

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

**U.S. MARINE SPECIFIC
SOFTWARE INTEROPERABILITY REQUIREMENTS
OF THE AFATDS AND IOS SOFTWARE SUITES**

by

Geoffrey D. Thome

September 2002

Co-Advisor:
Co-Advisor:

John Osmundson
Richard Riehle

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 2002	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE: U.S. Marine Specific Software Interoperability Requirements of the AFATDS and IOS Software Suites			5. FUNDING NUMBERS M9545002WRR2AMC	
6. AUTHOR(S) Geoffrey D. Thome				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Marine Corps Systems Command PG MAGTF C4ISR PMM 121 Ground C2 2033 Barnett Ave, Suite 315 Quantico, VA 22134-5010			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) <p>The Marine Corps has several Tactical Combat Systems at the Infantry Division level and below. The Information-Operations Server Version 1 (IOS v. 1) is a command and control (C2) system with a client-server architecture that when networked offers the Common Operational Picture (COP). The client is called Command and Control Personal Computer (C2PC). IOS was designed primarily to support maneuver, and has its roots in the Navy's Joint Maritime Command Information System (JMCIS). C2PC has been fielded to all Battalion and Squadron level and higher units in the Marine Corps, while IOS resides in Regimental and higher units.</p> <p>The Advanced Field Artillery Tactical Data System (AFATDS), originally designed by the Army, is the Marine fire support C2 System of Record. Current AFATDS software is tightly coupled to a particular hardware platform. AFATDS is currently being fielded to all units in the Fleet Marine forces.</p> <p>There are several problems with having two stand-alone C2 systems inside the same Combat Operations Center (COC). Among the most pressing problems is the inability for fires to support maneuver without tedious and dangerous manual conversion of data between systems. This thesis explores the software requirements for tactical systems integration of AFATDS and IOS.</p>				
14. SUBJECT TERMS Command and Control, Tactical C2 Systems, AFATDS, IOS, Intelligence-Operations Server, interoperability			15. NUMBER OF PAGES 151	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

**U.S. MARINE SPECIFIC
SOFTWARE INTEROPERABILITY REQUIREMENTS
OF THE AFATDS AND IOS SOFTWARE SUITES**

Geoffrey D. Thome
Major, United States Marine Corps
B.A., University of Colorado, 1990

Submitted in partial fulfillment of the
requirements for the degree of

MASTERS OF SCIENCE IN
INFORMATION TECHNOLOGY MANAGEMENT
And
SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2002

Author: Geoffrey D. Thome

Approved by: Professor John Osmundson
Co-Advisor

Professor Richard Riehle
Co-Advisor

Wayne Hughes
Dean, Graduate School of Operations and Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Marine Corps has several Tactical Combat Systems at the Infantry Division level and below. The Information-Operations Server Version 1 (IOS v. 1) is a command and control (C2) system with a client-server architecture that when networked offers the Common Operational Picture (COP). The client is called Command and Control Personal Computer (C2PC). IOS was designed primarily to support maneuver, and has its roots in the Navy's Joint Maritime Command Information System (JMCIS). C2PC has been fielded to all Battalion and Squadron level and higher units in the Marine Corps, while IOS resides in Regimental and higher units.

The Advanced Field Artillery Tactical Data System (AFATDS), originally designed by the Army, is the Marine fire support C2 System of Record. Current AFATDS software is tightly coupled to a particular hardware platform. AFATDS is currently being fielded to all units in the Fleet Marine forces.

There are several problems with having two stand-alone C2 systems inside the same Combat Operations Center (COC). Among the most pressing problems is the inability for fires to support maneuver without tedious and dangerous manual conversion of data between systems. This thesis explores the software requirements for tactical systems integration of AFATDS and IOS.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	PURPOSE.....	1
B.	BACKGROUND.....	1
C.	RESEARCH QUESTIONS	3
D.	SCOPE.....	3
E.	METHODOLOGY.....	4
	1. Research Methodology.....	4
	2. Interoperability Analysis Methodology.....	5
F.	ORGANIZATION.....	5
G.	BENEFITS OF THE STUDY.....	6
II.	MARINE CORPS AUTOMATED TACTICAL COMMAND AND CONTROL.....	7
A.	WHAT IS COMMAND AND CONTROL (C2)?.....	7
	1. Nature of C2.....	7
	2. People.....	8
	3. Information	9
	4. Support.....	11
	5. What Makes Effective C2	12
	a. <i>Command</i>	12
	b. <i>Control</i>	12
B.	C2 AT THE STRATEGIC AND OPERATIONAL LEVELS OF WAR.....	13
	1. DoD Strategic and Operational Processes	13
	2. Systems	14
C.	FACTORS AFFECTING TACTICAL C2	15
	1. USMC Maneuver Warfare Doctrine	15
	2. Implications of Combined Arms Concepts for C2	16
	3. Effects of Weapons Technology on C2	16
D.	BENEFITS OF AUTOMATED C2 SYSTEMS.....	17
E.	USMC C2 SYSTEM ARCHITECTURES.....	19
	1. Marine Corps View of Systems Architecture	19
	2. Problems with the USMC's Systems Architecture and Acquisition	21
	a. <i>Cultural Inertia</i>	21
	b. <i>USMC as System Buyer</i>	22
	c. <i>Stovepiped and Overlapping Development Efforts</i>	23
	d. <i>USMC Seen as a Bit Player Among Competing Interests</i>	24
	e. <i>Software Project Management Challenges</i>	24
F.	CURRENT C2 DEVELOPMENT ENVIRONMENT	25
	1. Overview	25

2.	DISA	26
3.	ASD, AT&L (Interoperability)	27
4.	GIG COE	27
5.	JROC	29
6.	SPAWAR Charleston.....	30
III.	THE UNIFIED MODELING LANGUAGE.....	33
A.	INTRODUCTION.....	33
1.	Object Orientation	34
2.	Advantages of UML	35
3.	Why UML is Useful for this Project.....	36
B.	USE CASES	36
1.	Actors.....	37
2.	Classes	38
C.	COLLABORATIONS.....	39
D.	CONCLUSION.....	41
IV.	THE INTELLIGENCE – OPERATIONS SERVER (IOS) SOFTWARE SUITE	43
A.	HISTORY	43
1.	Introduction	43
2.	The Maritime Command and Control Environment.....	43
3.	Joint Maritime Command Information System (JMCIS).....	45
4.	Tactical Combat Operations (TCO).....	47
5.	Intelligence-Operations Server (IOS).....	49
B.	OPERATIONAL REQUIREMENTS	50
1.	Selected User Requirements.....	51
2.	Actors.....	52
3.	Essential Use Cases.....	53
4.	COP Network.....	56
5.	Derived IOS Classes.....	58
6.	Interoperability Requirements.....	60
C.	IOS IMPEMENTATION ISSUES.....	61
1.	Software Development.....	61
2.	Support to the Fleet Marine Forces.....	62
3.	Future Capabilities.....	62
V.	THE ADVANCED FIELD ARTILLERY TACTICAL DATA SYSTEM (AFATDS).....	65
A.	HISTORY	65
1.	The Gunnery Problem	65
2.	Field Artillery Digital Automatic Computer (FADAC).....	66
3.	TACFIRE	67
4.	BCS/LTACFIRE/IFSAS.....	68
5.	AFATDS.....	69
B.	OPERATIONAL REQUIREMENTS	71
1.	Selected User Requirements.....	71

2.	Actors.....	73
3.	Essential Use Cases.....	74
4.	AFATDS Network	81
5.	Derived AFATDS Classes	82
6.	Use Case “Conduct Fire Mission” Sequence Diagram	84
C.	INTEROPERABILITY REQUIREMENTS	86
1.	Overview	86
2.	Interface Control Documents	87
3.	DII-COE Compliance Requirements	87
D.	AFATDS IMPLEMENTATION ISSUES	88
1.	The Master Unit List.....	88
2.	Current Capabilities	89
VI.	INTEROPERABILITY REQUIREMENTS ANALYSIS	91
A.	INTRODUCTION	91
B.	LISI-IMM	92
C.	USER REQUIREMENTS	94
1.	Requirements.....	94
2.	Actors.....	95
3.	Provide COP	96
4.	Use Case Conduct Fire Mission	98
D.	THE CRITICAL INTEGRATION CHALLENGE	102
1.	The AFATDS MUL.....	102
2.	Friendly Firing Unit Versus Track.....	103
3.	Unit Identification	103
4.	COP Object Translation.....	103
VII.	CURRENT INITIATIVES	105
A.	FIRE SUPPORT CLIENT	105
B.	PROXY SERVER	106
C.	TESTING	106
D.	FLEET SUPPORT CONTRACTORS.....	108
E.	VERSION CONTROL	108
VIII.	CONCLUSIONS.....	111
A.	GENERAL	111
B.	JOINT RECOMMENDATIONS.....	112
C.	MARINE CORPS RECOMMENDATIONS.....	112
1.	System Architecture Management	112
2.	MARCORSYSCOM	113
3.	Fleet Contractors.....	114
4.	MCTSSA	114
D.	RECOMMENDATIONS FOR NPS.....	115
E.	THE UML AS A MODELING TOOL.....	116
F.	AREAS FOR FURTHER RESEARCH	116
	LIST OF REFERENCES	119

APPENDIX A. GLOSSARY	125
APPENDIX B. EXAMPLE MARINE ORGANIZATION FOR COMBAT	129
INITIAL DISTRIBUTION LIST	133

LIST OF FIGURES

Figure 1.	The OODA Loop (From [MCDP06, 64])	9
Figure 2.	The Information Hierarchy (From [MCDP06, 67])	10
Figure 3.	Typical COP Display (Author's files).....	18
Figure 4.	The DII COE Conceptual Model (After [WALK01, 3]).....	28
Figure 5.	Unit Tracking System.....	38
Figure 6.	UML Example Class Diagram	39
Figure 7.	UML Collaboration Diagram	40
Figure 8.	Example Sequence Diagram	41
Figure 9.	JMCIS Lineage (After [BUDD02]).....	46
Figure 10.	IOS in a tactical environment.....	49
Figure 11.	IOS Essential Use Cases	53
Figure 12.	Regimental COP Network.....	57
Figure 13.	IOS COP.....	58
Figure 14.	IOS Track Class.	59
Figure 15.	FADAC terminal. (From [BRLA61, 254])	67
Figure 16.	AFATDS Workstation (From [APIC02])	71
Figure 17.	AFATDS System Level Use Cases.....	75
Figure 18.	Example Infantry Regimental AFATDS logical Network.....	81
Figure 19.	AFATDS OPFAC Class.....	83
Figure 20.	AFATDS Provide COP	83
Figure 21.	AFATDS Guidance Classes.....	84
Figure 22.	Use Case "Conduct Fire Mission" Sequence Diagram	85
Figure 23.	IOS Client Call For Fire Sequence Diagram.....	101
Figure 24.	Marine Artillery Battalion in Direct Support of an Infantry Regiment.....	129
Figure 25.	Example Regimental COC. (After [LITT02]).....	130

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1.	“One Architecture, Three Views.” (After [MSEI02B, 4])	20
Table 2.	DII COE Levels of Compliance (After [DIRS97].)	29
Table 3.	Pertinent C2 Projects at SPAWAR Charleston	31
Table 4.	Unit Tracking System Requirements	36
Table 5.	Unit Tracking System Use Case.....	37
Table 6.	Selected IOS User Requirements (After [TORD95], [TCOE95])	51
Table 7.	IOS Actors.....	52
Table 8.	Essential Use Case Provide COP (IOS).....	55
Table 9.	Essential Use Case Command Forces (IOS)	56
Table 10.	IOS Track types. (After [CHBK98, 24-26]).	60
Table 11.	AFATDS User Requirements (After [FSSS00]).....	73
Table 12.	AFATDS Actors.....	74
Table 13.	Essential Use Case - Provide COP (AFATDS).....	77
Table 14.	Essential Use Case – Define Fire Mission Criteria. (AFATDS).....	79
Table 15.	Essential Use Case – Conduct Fire Mission (AFATDS)	80
Table 16.	LISI-IMM Attributes (After [LISI98, 2-8]).	93
Table 17.	IOS/AFATDS Federation Actors.	96
Table 18.	IOS Client Call for Fire Use Case.....	100

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I have been fortunate to have the support of about a hundred outstanding American military and civilian professionals, without which this thesis would not have been possible. First, I extend my sincere appreciation to Captain Adam Kubicki, recently departed from the AFATDS Project Office at the Marine Corps Systems Command, for his support and his willingness to provide research funding from the AFATDS account. Second, Professors Osmundson and Riehle provided invaluable assistance and guidance in the development of the many process and software concepts contained herein. Third, Lieutenant Colonel Stan Watkins, USMC (Ret) provided critical input on the rough draft, ensuring that history was properly recorded and future concepts were accurately described. Finally, I would like to thank my wife AnneMarie, for her patience and understanding throughout this research effort. What errors are contained in this document are purely mine.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PURPOSE

Interoperability is defined as:

The ability of systems, units or forces to provide data, information, materiel, and services to and accept the same from other systems, units, or forces and to use the data, information, materiel, and services so exchanged to enable them to operate effectively together. IT interoperability includes both the technical exchange of information and the end-to-end operational effectiveness of that exchange of information as required for mission accomplishment. [DODD02, 13]

The purpose of this thesis is to investigate some of the reasons why Command and Control Systems interoperability is so difficult to achieve and to offer both short term and long-term solutions. The military has focused on interoperability via mandated technical architectures while ignoring the fact that the logic embedded in system software is the primary source of roadblocks to system interoperability. For the DoD, it is simpler to focus on technical standards than to get any of the various services or subspecialties to agree on exactly what each C2 system should do. Due to its unique nature as “Soldiers from the Sea,” the Corps interfaces directly with each service, and often buys its systems from the various services. It therefore bears the brunt of many inter-service interoperability problems. This thesis will focus on the interoperability requirements for two current Marine Corps ground command and control systems, the Intelligence-Operations Server (IOS), and the Advanced Field Artillery Control System (AFATDS).

B. BACKGROUND

In keeping with the American tendency to always seek a better way to do something, the Department of Defense has continually sought to better control its forces through the use of computer and other technologies. The intended benefits of automated control were the ability to control larger forces over a wider area, gain more and better intelligence, and speed decision-making. One early example of military automation was the use of the ENIAC computer to calculate firing tables for the field artillery during the Second World War. However, then as now there were significant social and cultural

barriers between military and the computer developers. Because of these barriers the military focused on physical hardware characteristics, while the logical and software designs were left to the white coated people - others with more interest in these areas. Further, the military was often balkanized along services and functional specialties, and regularly failed to coordinate parallel weapons development and acquisition. In the automated Command and Control arena, the result is a hodgepodge of incompatible hardware and software solutions. Some systems effectively manage a particular function of C2, while others serve as bad examples of wasted funds and overblown expectations. Although the Global Command and Control System (GCCS) is a successful replacement for the World-Wide Military Command and Control System (WMCCS) at the strategic and operational level of war, there is currently no integrated C2 solution for the tactical warfighter.

The Corps focuses on winning battles and making Marines (i.e., instilling Marine Corps values into recruits) and generally operates at the operational and tactical levels of warfare. The Marine Corps rightfully prides itself on its long and illustrious history of combat success. As the smallest service in the DoD, it has traditionally been concerned with its own funding and survival. To these ends, it has justified itself to the American people by being the best at every assigned mission. Although the Marine Corps is innovative (for example, it developed effective amphibious tactics well before World War II required them), the Corps relies on tried and true methods and equipment. The Corps doesn't have a lot of patience for ineffective or complicated systems. Only in the last few years have Marines started to realize the potential benefits of reliable, deployable Command and Control Systems.

Finally, the Corps traditionally is an equipment buyer, rather than an equipment developer. The Corps does not have large research and development budgets, and what budget the Corps does have is spent on developing transformational technologies such as the MV-22 and the Advanced Assault Amphibian Vehicle (AAAV). The Corps allows the other services to develop equipment as a risk-mitigation strategy. The other service assumes the brunt of developmental costs and failure risk, while the Corps buys the resulting equipment. Examples of this strategy abound, and include everything from the

M-16 to the F-18. The result is that some Marine-specific requirements often are left out of new products, because the Corps bought the equipment off the “government shelf.”

Despite these impediments, the Corps has a significant need for effective ground C2 systems. The benefits of automated C2 are primarily in speedier decision-making and greater span of control. The Corps needs innovative solutions to C2, and it needs to be able to interoperate with other joint and coalition forces. The ultimate goal of Marine C2 is to better accomplish the Marine warfighting mission while limiting resource use and minimizing casualties.

The Marine Corps Systems Command (the organization responsible for development and acquisition of Marine Corps systems and equipment) has no less than 30 different Command and Control Systems under development. The Ground C2 directorate has seven of these programs under its cognizance, of which two are the IOS and AFATDS. IOS manipulates and displays the Common Operational Picture (COP), while AFATDS manages fire support. Both these systems are in wide use in the Marine Corps today. The lack of interoperability between these two systems is a critical issue because discrepancies between the two systems contribute to lack of trust in the information presented to the Commander and degrade the utility of both systems.

C. RESEARCH QUESTIONS

This research addresses the following primary questions:

1. What are the logical underpinnings of AFATDS and IOS?
2. What are the software interoperability requirements between these two systems?
3. What near-term solutions are there to the interoperability problem?
4. What long-term solutions will be effective in mitigating or preventing future interoperability issues?

D. SCOPE

Interoperability and integration are recognized both inside and outside DoD as critical issues. Because of this, there is a significant body of literature and great interest in the topic. This thesis will narrow the topic by focusing on the logical (high level or

system level) interoperability and integration issues between these two C2 Systems. Further, it will focus on the two primary tasks of maintaining the Common Operational Picture (COP), and calling for fire support. Interesting requirements are requirements about items that help build the COP, such as unit information, position-location information, fire mission data, and Commander's Critical Information Requirements. Uninteresting requirements are for peripheral and supporting data, formatting data, and textual messages. Also, although there are significant interoperability issues at the hardware and joint technical architecture levels (i.e., do all systems work together on an Ethernet LAN or use compatible protocols), these issues are not covered in this thesis.

E. METHODOLOGY

1. Research Methodology

The research methodology consisted of literature reviews of both government and commercial industry documents, personal interviews with program managers, contractors and users, and visits to operating units, testers, and developers. Financial support and extensive research material was received from the AFATDS Program Office, Marine Corps Systems Command, Quantico, Virginia.

In order to better understand the specific interoperability problems, a trip was taken to 29 Palms, California to observe Combined Arms Exercise CAX 3 and 4, 2002. The unit undergoing training at these exercises was MAGTF-6, under the leadership of Colonel John Coleman, Commanding Officer of 6th Marine Regiment, 2nd Marine Division. During the period of his command, Colonel Coleman took a proactive approach to the use of digitized C2, and developed techniques and procedures to effectively use currently fielded systems. His primary focus has been on the effective use of the COP at the Regiment and below.

The second field trip was taken to the Marine Corps Tactical Systems Support Activity (MCTSSA), Camp Pendleton, California, to observe IOS/AFATDS interoperability testing. This testing was conducted from 25 Feb – 1 Mar 2002 in response to a message from the First Marine Expeditionary Force (I MEF) Fires Section,

indicating serious operational deficiencies and possible fratricide issues in the interface between these two systems. [1MEF02]

2. Interoperability Analysis Methodology

Joint and Marine Corps Command and Control doctrine is used as justification of interoperability requirements. C2 Systems are complicated. Both systems under study were developed for different customers using differing requirements and under widely differing conditions. Any comparative study of both systems using the requirements development method from one system would be automatically biased against the other. For this reason, the Unified Modeling Language (UML) was chosen to model the significant aspects of both systems. UML provides a common visual language for representing software systems.

Interoperability requirements and recommendations will be presented in the context of the Levels of Information System Interoperability – Interoperability Maturity Model (LISI-IMM). [LISI98] This model succinctly states the requirements for complete system interoperability, and provides a five-point scale for objectively evaluating current systems interoperability. Although there is much discussion about the effectiveness of the automated tools used to evaluate Programs of Record under the model, the model's conceptual framework is sound.

F. ORGANIZATION

This thesis is divided into seven chapters, which are organized as follows:

Chapter II introduces the concepts of Command and Control and outlines Marine Command and Control doctrine. It outlines some of the arguments for and against C2 automation. This doctrinal underpinning justifies the requirements for automated command and control.

Chapter III introduces the basic concepts of object orientation and describes pertinent features of the Unified Modeling Language. Justification is provided as to why UML and object orientation is appropriate for this interoperability discussion.

Chapter IV describes the Intelligence-Operations Server. It outlines the history of the system, analyzes system and interoperability requirements, and describes pertinent current system capabilities.

Chapter V describes the Advanced Field Artillery Tactical Data System. It outlines the history of fires automation and the genesis of the system, analyzes system and interoperability requirements, and describes pertinent current system capabilities.

Chapter VI is an analysis of the interoperability needs of these two systems in light of the warfighter's need for effective command and control.

Chapter VII discusses current initiatives that various organizations are taking to increase the interoperability of these two systems.

Chapter VIII lists the conclusions and recommendations of the study.

G. BENEFITS OF THE STUDY

The primary benefit of this study is a better understanding of the challenges facing the Marine Corps in the acquisition, integration, and use of disparate Command and Control systems. Due to its size and primary warfighting focus, the Marine Corps is in the position of system buyer, rather than system developer. Furthermore, due to the unique nature of Marines as "Soldiers of the Sea," the Marine Corps is on the leading edge of inter-service interaction. In the Command and Control domain, this interaction typically plays out in serious interoperability disconnects between joint systems while the Marines are performing real-world missions. Therefore, it is imperative that the Marine Corps accurately and forcefully articulate its requirements for interoperability to the DoD and other services while they are building the systems the Marines will someday use.

II. MARINE CORPS AUTOMATED TACTICAL COMMAND AND CONTROL

A. WHAT IS COMMAND AND CONTROL (C2)?

The joint definition of Command and Control (C2) states:

Command and Control is the exercise of authority and direction by a properly designated commander over assigned and attached forces in the accomplishment of the mission. Command and Control functions are performed through an arrangement of personnel, equipment, communications, facilities, and procedures employed by a commander in planning, directing, coordinating, and controlling forces and operations in the accomplishment of the mission. [JPUB01, 89]

Effective C2 is the critical element in the success of every military mission. C2 is a process. Computer systems designers must understand this process prior to system design. C2 doctrine forms the basis of all C2 system requirements and merits further study.

1. Nature of C2

The Marine Corps considers C2 to be “the means by which a commander recognizes what needs to be done and sees to it that appropriate actions are taken.” [MCDP06] From this simple definition one can see that C2 frames every decision the military takes. Command and Control is a pervasive process rather than one among a set of warfare specialties. C2 is also highly personality-dependent, as it is the commander who exercises C2. C2 is not assumed but rather C2 is assigned by proper authority. Command and control has two components.

The first component, Command, is the exercise of properly assigned authority over designated forces. It is seen as flowing from the commander to the commanded forces. Command is the use of the commander’s will to carry out tasks. Command may be limited by statute, mission, or situation. The second element, control is generally seen as representing the methods that the commander takes to enforce command. However, the Marine Corps takes the view that control serves as feedback to the commander,

carrying information about the command function. [MCDP06, 41] In either case, both command and control are descriptions of social interaction. While the joint definition of C2 mentions the five components of personnel, equipment, communications, facilities, and procedures, the Marine Corps reduces these items to the three “Pillars of Command and Control.” These pillars are: people, information, and support.

2. People

The first pillar of C2 is people. Commanders are people commanding other people who may themselves be commanders. Modern warfare precludes the commander from personally observing everything on the battlefield, and therefore people provide information to the commander. Further, the commander is unable to personally control all assigned forces, and so people (staff and other commanders) help the commander make decisions and take action. The measure of the size of forces a Commander can manage is called “Span of Control.” Automation increases the Span of Control.

As an action of people, command is inseparable from leadership. The art of leadership is the art of people making decisions. People, not computers, make decisions. Marine Corps doctrine represents the process of coming to a decision using John Boyd’s Observe-Orient-Decide-Act (OODA) loop (Figure 1). As shown in the figure, this loop is a continuous cycle. First, the commander must observe his environment and internal situation. Then the commander orients on the militarily important aspects of the situation. Using a formal decision-making process, his own intuition, or a combination, he comes to a point of decision. He then acts on that decision in order to defeat the adversary. The mechanics of the loop may also be applied to military organizations. It is generally accepted that the side that more quickly executes the loop will have the advantage in any conflict. The promise of automation is that the OODA Loop can be executed exponentially faster than manual methods for large Spans of Control. Because Command and Control is a function of people, Marines are wary of systems that try to remove decisionmakers from the Command and Control process.

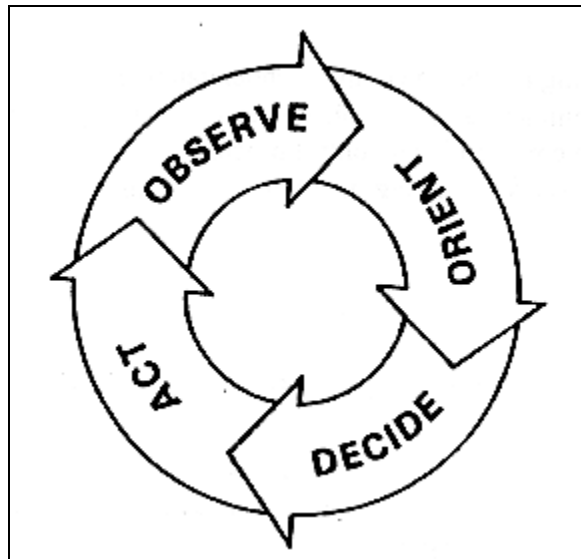


Figure 1. The OODA Loop (From [MCDP06, 64])

Organization and training are two final aspects of people worth mention. People are organized by the commander to meet the mission. A poor organization will slow the collection of information, complicate the decision-making process, and take insufficient or inappropriate action. The commander therefore sets the organization and trains his people to accomplish the mission and meld his people to his decision-making style.

3. Information

The second pillar of C2 is information. Information is generally considered in terms of an information hierarchy. The Marine Corps uses the hierarchy shown in Figure 2 below. Each level up in the hierarchy requires processing effort and filtering. In this hierarchy, understanding is the highest rung, and occurs in the minds of people after the processing of information with their own judgment. Data is the lowest rung in the hierarchy. Without processing, data is useless. Instead, data appropriate to the situation must be collected, analyzed, fused with other data. Processed data must be analyzed to produce knowledge. Then this knowledge must be formatted and presented to the commander in a timely manner in order to be militarily useful. Because of the sea of data available, the commander risks information overload. Therefore he must focus his intelligence (“data”) gathering processes to collect only critical information. The Commander’s Critical Information Requirements (CCIR’s) are an educated guess on

what will be the information most relevant to making appropriate decisions, given the current situation.

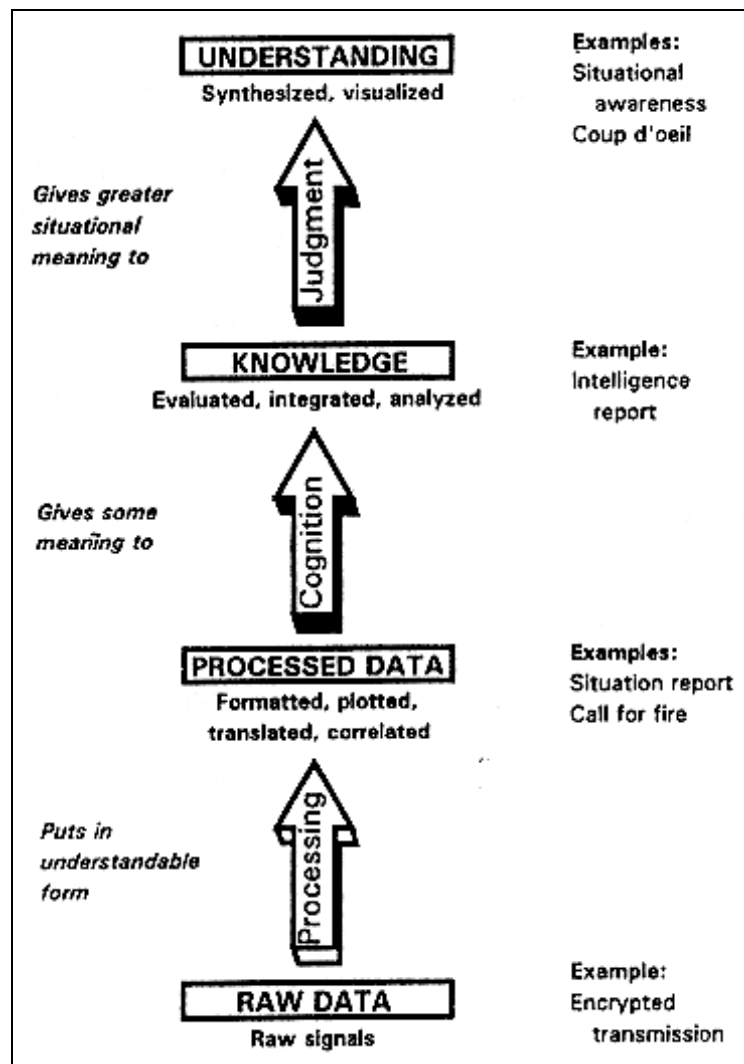


Figure 2. The Information Hierarchy (From [MCDP06, 67])

As with people and the OODA loop, the information hierarchy benefits from functional, reliable automation. Information in the military is time-critical, as information feeds the OODA loop. Information must be available when the commander needs it. The manner in which information is presented is also critical to absorption. Most people are more responsive to pictures than text. Automated systems must therefore be able to present the complexities of the military situation pictorially and graphically.

In conflict, information needs are never completely filled. Because the warfighting environment is so chaotic, a certain level of information deficit must be accepted. Many times, the requirement to decide and act swiftly and forcefully outweighs the requirement for more information. Therefore, automated C2 systems must focus on satisfying requirements over maximizing requirements. Automated systems must add value to the commander despite an atmosphere of incomplete or partially inaccurate information.

4. Support

The final element of Command and Control is support. Support includes items listed in the Joint definition of C2: the equipment, communications, facilities, and procedures. Support includes all those things that compose an automated system, including the communications network, specialized hardware, and software. Support always has been an element of C2 – an example from the Civil War era is the use of the telegraph to speed C2 functions. The telegraph was an element of information grid of the Civil War Era.

Marine doctrine emphasizes that people and information are much more important than any element of support. This emphasis is added to ensure that technological wizardry never takes the place of the commander's decision-making authority. There is really no concern in the case of procedures – the Corps controls procedures. However, “technology” is another matter. The Corps views talk from other services of hi-tech decision-making systems with skepticism. The Corps also rejects technology that permits micromanagement of tactical commanders. Instead, the Corps emphasizes that technology should enhance performance while allowing freedom of action for the commander. Technology may reduce the number of people that are in the organizational chain, but Marines are wary of decisions made by Command and Control systems. This particular issue is explored more fully in the AFATDS chapter.

5. What Makes Effective C2

a. Command

Command is effective when the commander's intent is clearly stated. The commander's intent is a short statement on why a particular action is being taken and what the desired end state of the action is. In a rapidly changing situation or the absence of communications as may occur from time to time in war, a clear commander's intent allows for independent action of subordinates to achieve the common goal.

Command is effective when all parts of the military community plan collaboratively with a single goal in mind. This single goal is referred to as the "single battle." Once the single battle is agreed upon, the various parts of the military can plan with a minimum of confusion and friction. The modern military is a complicated system, and lack of cohesion can result in fratricide, confusion, and even mission failure.

Command requires unhindered communication both up and down the chain of command. Communication is the sending and receipt of messages. Both the message processing and the content must be standardized between parties. For spoken communication between the services, this clarity starts with an agreed-upon language. The Joint Dictionary of Terms [JPUB01] provides that common language to the warfighter. Other agreed-upon standards provide the equivalent support for automated systems.

Command requires the appropriate level of supervision to ensure that the commander's intent is carried out to completion. As previously mentioned, the Corps is concerned about technology being misused as an enabler for micromanagement. However, an appropriate level of supervision must be supported by automated systems, and it is incumbent upon leaders to focus on their responsibilities, even when an automated system may appear to provide the ability to control everything.

b. Control

In Marine doctrine, control is represented as feedback from lower echelons to the commander. [MCDP06, 40-41] The commander therefore receives

control from the force. Taken in this light, effective control means the Commander receives critical information in time to make an appropriate decision (i.e., a decision as nearly correct as possible given the situation). Control implies robust duplex (two-way) communication. Control requires low latency. Control requires accuracy. Finally, control includes those actions taken to support the commander's decision-making process.

B. C2 AT THE STRATEGIC AND OPERATIONAL LEVELS OF WAR

1. DoD Strategic and Operational Processes

As a result of several serious operational deficiencies in the Department of Defense during and prior to the 1980's, the Department of Defense was reorganized in 1986 by the Goldwater-Nichols act. The intent of this legislation was to force the U.S. military to fight jointly. The benefits of fighting jointly are less interservice rivalry, reduction in force duplication, clarified service roles, a streamlined joint command structure, better advice for the president and civilian authorities, reduced costs, and fewer casualties from internal friction. The performance of the DoD in Desert Storm served to validate the concepts embodied in the Goldwater-Nichols legislation. This legislation has had lasting positive effects on each service and the Department of Defense, and is considered a success.

The Marine Corps and deployed Marine units generally are employed at the tactical and operational levels of war, although actions taken at this level of warfare can have strategic as well as tactical and operational consequences. Marine forces are integrated under the Unified Commands with the other services in accordance with the Unified Campaign Plan (UCP). The UCP divides the globe into regions and assigns responsibility for military affairs to Unified Commanders in each region. An example regional command is the United European Command (EUCOM), which is responsible for American military forces in Europe and surrounding areas. There are presently 6 Unified Combatant Regional Commands, including the new Northern Command.

The DoD has standardized its planning and warfighting processes to meet the requirements of joint, global action. The primary process the Joint community uses is the Joint Operational Planning and Execution System (JOPES). JOPES is an agreed upon language and a standardized set of processes, supported by common automated systems. [JJPS95, i] Primary JOPES products are campaign plans and force deployment information. This critical planning function allows the military to prepare for contingencies. JOPES processes can be used deliberately or in response to a crisis. JOPES products (the Operations Plans (OPLANS) and force deployment data (TPFDD)) are information intensive.

2. Systems

The JOPES process was first supported by the World-Wide Military Command and Control System (WMCCS). The Global Command and Control System (GCCS) replaced WMCCS in the mid-1990's. GCCS incorporates the core planning and assessment tools required by the joint community to meet the information requirements of the services. GCCS is composed of several mission applications (called the Unified Build) built on a single Common Operating Environment (COE). GCCS is designed for networking. GCCS maintains the Common Operational Picture (COP) for the President of the United States, the Secretary of Defense, and the Unified Commanders. GCCS primarily serves the strategic and operational levels of war.

Because the modern military fights jointly and often with coalitions, Command and Control systems at all levels have much greater demands for interoperability than would be expected for systems that just support the services. Further, C2 systems must be able to operate across service, distance, and jurisdictional boundaries. As a practical matter, that means that all C2 systems must interface with GCCS.

C. FACTORS AFFECTING TACTICAL C2

1. USMC Maneuver Warfare Doctrine

The Marine Corps has adopted “Maneuver Warfare” for its fighting doctrine. The alternate choice is “Attrition Warfare.” Attrition Warfare defeats the enemy by head-to-head engagement with the opposition. In these engagements, the enemy is reduced by whatever means available, usually with the overwhelming application of combined arms. The primary focus often is reduction of the adversary’s military force. Attrition warfare is characterized by the extensive use of fires to reduce the enemy in frontal assaults. The command structure for Attrition Warfare can be rigidly hierarchical, because what matters is the ratio of friendly to enemy combat power. Since Attrition Warfare masses combat power, lower levels of leadership are within the senior commander’s span of control, and may have little decision-making authority.

Maneuver warfare, on the other hand, endeavors to find the enemy’s weaknesses and exploit these weaknesses to bring about the desired end state. These weaknesses may not be associated with the enemy’s combat power. Maneuver warfare may require the application of overwhelming combat power, but the combat power is applied to the selected weaknesses. The weaknesses that Maneuver Warfare focuses on are those that will bring the conflict to an end with a minimum of effort and casualties. These weaknesses are known as “Critical Vulnerabilities.” Maneuver Warfare requires more independent action on the part of subordinate leaders, because the senior commander may not know where all the enemy’s Critical Vulnerabilities are. Instead, the senior commander issues a mission-type order with his intent clearly stated, and expects his subordinates to act with judgment in order to meet the commander’s intent.

As a practical matter, automated C2 systems used to support Maneuver Warfare must support C2 decision-making at both higher and lower levels, while being able to present the same information to each level. Automated systems must facilitate information passing between each level of command while being robust enough to maintain the last known information if communications are lost between units.

2. Implications of Combined Arms Concepts for C2

Both attrition warfare and maneuver warfare require the use of “combined arms.” Combined Arms is the application of two or more warfighting techniques together in time and space on the enemy. Generally, combined arms uses a direct-fire method and an indirect fire method. As an example, consider an enemy defensive position. A combined arms attack could have tanks firing at the position while aircraft drop bombs on it. The adversary cannot respond to the tank fire without exposing himself to the effects of the aircraft, and cannot respond to the aircraft without exposing himself to the effects of the tank fire. The critical elements to making combined arms effective are the effectiveness of each individual arm, and the combination of them in time and space. Ensuring that each arm does not affect the other (i.e., avoiding fratricide) is called deconfliction.

The implications of combined arms for C2 systems are complex. First, C2 systems must be able to share information across platforms. C2 systems must be able to share information between the various warfighting communities with a common electronic data dictionary. In the example above, the aircraft must be able to communicate with the tank commander. The tank commander must know aircraft-specific terms, and the pilot must know all about ground terms. In order to assist with the deconfliction problem, C2 systems must know the physical effects of the ordnance being used. The C2 systems must be on common time and share common geo-spatial data. The C2 system must be able to operate in a time-constrained environment.

3. Effects of Weapons Technology on C2

As weapons have gotten more lethal, friendly force dispersion has grown. Greater dispersion means C2 systems must communicate across greater distances. A continuing problem with current C2 systems is their inability to communicate the required amount of information across empty space, when both senders and receivers are mobile.

One aspect of weapons lethality is precision strike. Precision strike is the ability to hit targets with a high degree of accuracy. Precision strike drives a need for better targeting information. The need for better targeting information drives the need for more

and better sensors. The net effect is to increase the amount of data passed between the sensor and the shooter, and the load on C2 systems.

D. BENEFITS OF AUTOMATED C2 SYSTEMS

Automated C2 systems have the capability to generate and maintain the Common Operational Picture (COP). An Operational Picture is a model of reality rather than a discrete thing. Conceptually, the Operational Picture is the shared awareness that the commander and his staff have of the current situation. The concept is formed, shared, and maintained by models the commander uses to represent the situation. For example, the Operational Picture can be established on a gridded sand table, with small tanks representing units on the ground. Or the Operational Picture can be represented on overlays to a map, with pushpins representing units and hand-drawn lines representing various boundaries. Finally, the Operational Picture can be a computerized display of information overlaid on a digital map. What differentiates the COP from an Operational Picture is that two or more commanders share the COP.

Having everyone use the Common Operational Picture is critical to modern warfighting. Even if some information in the COP is incorrect, having the same data across all commands allows for the subject matter expert to correct the mistake one time, and then share that correction with everyone else. The Joint definition of the COP is: “A single identical display of relevant information shared by more than one command.” [JPUB01, 86] As indicated in the definition, the COP adds this final component of commonality across commands. An example of a COP display is shown in figure 3 below.

One problem with the Joint definition is the use of the words “identical display.” COP displays are built out of the data stored in automated systems. The displays don’t need to be identical, but instead the underlying data must be identical. For instance, a maneuver unit may need a display tailored to present maneuver data, while a fire support unit may need a display tailored to present fires data. In Software Engineering, a different representation of the same item is known as a View. Therefore, a View corresponds with a “Display,” while the underlying data model must agree between systems.

Because of this and other subtle differences in the meaning of COP, two other terms that represent a portion of the COP are used. The first term, the Common Relevant Operational Picture (CROP), attempts to limit the amount of information that any one node must process, using time and space criteria (in other words, “old” data is irrelevant). The second term, the Common Tactical Picture (CTP), attempts to limit the amount of information that a node must process by limiting the scope of applicability. GCCS is the Joint System of Record for the COP, while subordinate systems may contain the CTP. In either the CTP or the CROP, the commander decides what is relevant, and appropriate filters are applied to the COP to generate these views. For the remainder of this thesis, the term COP is used over that of CROP or CTP, emphasizing the need for commonality across systems.

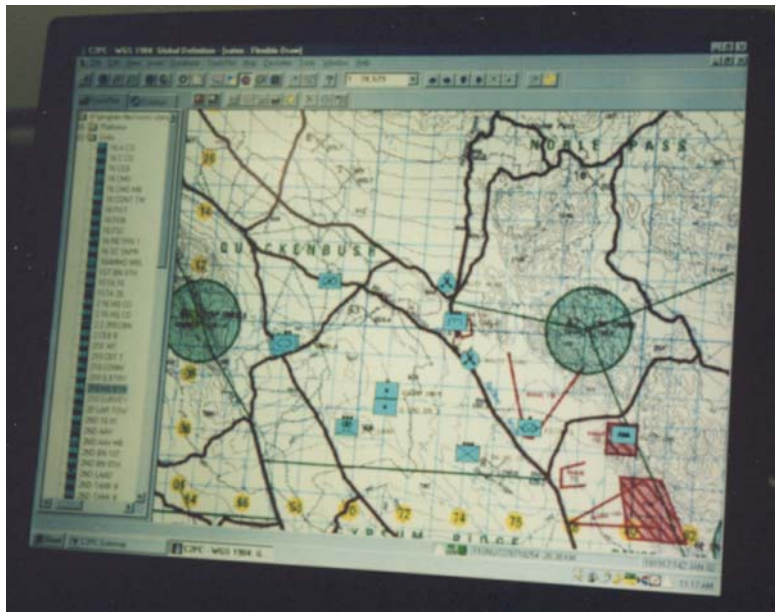


Figure 3. Typical COP Display (Author's files).

The ability of properly designed automated systems to share the masses of time-critical information the COP requires across wide distances is unmatched by any comparable manual process. The advantages automation promises are that the commander's span of control of forces can grow to support truly global operations. Commanders and staffs can share information and collaborate faster. Automation permits the staff to clear up some of the confusion of battle. Automating other Command and Control functions (such as fires delivery in the case of AFATDS) provides similar

benefits for those functional areas. Most importantly, automation allows us to execute the OODA cycle faster than the enemy. Rapid Decisive Operations (RDO) is the DoD buzzword for this quantum leap in the ability to quickly act.

E. USMC C2 SYSTEM ARCHITECTURES

1. Marine Corps View of Systems Architecture

Generically, Systems Architecture is defined as “Design. The way components fit together.” [IEEE90] For the Marine Corps, the Systems Architecture is the plan by which automated systems and weapons are designed, developed, and procured in order to fit together with other systems. A well-designed and executed Systems Architecture will provide the framework for effective Command and Control automation across disciplines. Where there is more than one system, there is a de-facto Systems Architecture. If the architecture is ad-hoc, and systems are bought piecemeal, the architecture will be poor. The important thing is to pre-plan systems development and acquisition so that the Systems Architecture that results is the architecture that supports the operational requirement. The operational requirement is often expressed as “Operational Architecture.”

The Marine Corps Systems Command (MARCORSYSCOM) is the Marine Corps’ principal agent for equipping the operating forces to accomplish their mission. They are responsible for developing, acquiring, and fielding automated systems. The Systems Engineering and Integration (SE&I) branch of MARCORSYSCOM is responsible for the Marine Corps C2 systems architecture. Specifically, SE & I is chartered to:

- Create and Maintain the Marine Corps C4ISR¹ Systems Architecture
- Provide macro-level configuration management for C4ISR systems within the MARCORSYSCOM
- Provide interoperability and integration analysis for C4ISR systems within the MARCORSYSCOM [MSEI02A]

¹ C4ISR stands for “Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance.” As previously discussed, the term “Command and Control” already contains these added elements. However, variant terms like C3, C4, C4I, and C4ISR are common in the literature.

SE & I's view of the Marine Corps C2 architecture is shown in the table below.

	Technical Architecture	Systems Architecture	Operational Architecture
Controlling Authority	Headquarters Marine Corps	MARCORSYSCOM	Marine Corps Combat Development Command
Domain	Standards, Protocols, Interfaces	Systems and their relationships (Components and Subcomponents)	Organizations, Functions, & Information exchange (Concepts)
MARCORSYSCOM Role	Use the JTA ² to build the Systems Architecture	Create & maintain the Systems Architecture	Use analytic tools to assess where a given system “fits” into the Systems Architecture
“MAGTF City” Example	Plumbing and Electrical codes of house	City Map with Neighborhood & Street Location	City Zoning: location supports activity

Table 1. “One Architecture, Three Views.” (After [MSEI02B, 4])

Creating good Systems Architecture from the Ad-Hoc architecture of legacy systems is a tall order. In order to scope the problem, SE&I has built a database (called MSTAR, discussed in chapter 7) that accurately describes the current systems. From the database, they have created the “MAGTF C4ISR Integrated Picture (MCIAP).” The MCIAP pictorially represents all currently fielded C2 systems at the echelon of command that each system is used. It further shows the communications links by frequency spectrum (or LAN as applicable) between each node in the command chain. The MCIAP had to be 13 feet long by 3 feet high in order to accurately and legibly list all the current Marine C2 systems. This alone shows the complexity of the problem of system-of-systems integration. While one can argue whether the architecture depicted on the MCIAP is planned or Ad-Hoc, the fact remains that current C2 systems lack the interoperability to meet the needs of the FMF.

² JTA stands for “Joint Technical Architecture.” The JTA is a collection of interface and other technical standards mandated by the Defense Information Systems Agency, and is discussed further later.

SE&I is taking steps to better define and refine the C4ISR systems architecture. First, they are taking the approved “C4I Support Plan” (C4ISP – see the section on the Defense Information Systems Agency below) from the joint community and applying it to future Marine Corps system buys. Second, the SE&I has developed a problem targeting process for allocating limited funds. This process conducts “problem triage” by dividing problems into near-term, mid-term, and far-term interoperability issues. Near-term issues are defined as those issues that must be handled before the next budgeting cycle, while mid-term issues are handled in the next budgeting cycle. Finally, far-term issues are those that occur beyond the next budgeting cycle. The choice of the budget cycle as a delimiter is a natural one for MARCORSYSCOM, as the budgeting cycle provides the capital required to make changes. However, the budgeting cycle is complex and maddeningly slow. Generally, Marines in the FMF will not see the results of a “mid-term” fix for a minimum of 4 years from the identification of a problem.

No matter the timeframe of the fix, SE&I is able to bring pressure to bear on Marine Program Officers to modify requirements to meet interoperability goals. These program officers also work at MARCORSYSCOM, and each of their programs have “Integration Key Performance Parameters” (I-KPP’s) that contractors must meet. SE&I’s intent is to better define these I-KPP’s organizationally, and then require the Program officers to force contractors meet them. Finally, SE&I branch has all C2 systems under configuration management, and has tasked MCTSSA with conducting end-to-end systems testing. These efforts are explored further in Chapter 7.

2. Problems with the USMC’s Systems Architecture and Acquisition

Myriad items prevent reaching the goal of true systems interoperability. The list of items below is a short explanation of the issues most important to the Marine Corps interoperability problem.

a. Cultural Inertia

Institutionally the Marine Corps has embraced C2 automation technology as a warfighting enabler. The rank and file is not so sure. As previously mentioned, the

Corps culture is resistant to change. This list is a representative sample of comments about C2 heard from FMF commanders and staffs:

- “A computer with a bullet hole in it is a paperweight. A map with a bullet hole in it is still a map.”
- “I’d rather command from the front, not from behind a computer.”
- “So now we’re bringing five generators and three dozen computers?”
- “What’s wrong with voice? Digital is too hard.”

The first comment, most recently overheard from a Marine Colonel in May 2002, reflects distrust in C2 systems resulting from a perceived lack of system availability in combat conditions. The second comment addresses the position that C2 systems are for a different (lower) class of people than warfighters and couches the argument in terms of leadership. The third comment reflects the FMF’s experiences with unwieldy legacy systems that use excessive valuable combat lift. The fourth comment points to the need for specialized skills to operate automated systems.

None of these comments bears up under close scrutiny, but insofar as Marines feel this way, they tend to act in ways that fulfill their perceptions. For example, a Commander that doesn’t trust C2 systems won’t use them. If the systems aren’t used, the specialized skills that are needed to operate them atrophy. Without use, the systems can’t be improved. Any spare funds that units receive is not spent on improving automated gear, but instead is spent in other ways. The net result is the status quo is maintained to the ultimate detriment of the organization.

b. USMC as System Buyer

The Marine Corps is most often a system buyer rather than a system developer. [MSEI02C, 3] The Marine Corps operates this way because it saves costs and lowers risks on individual systems. Another service, as lead developer, must manage the program and bear the associated costs. The lead service must answer questions from congress and other oversight bodies about cost, schedule, and performance issues. Risks for the Marine Corps are lowered because at the end of the process, the Marine Corps gets the finished product. Other services like the arrangement because having the Corps participate in a project allows for economies of scale, and the item can be marketed as a multi-service or joint product.

Yet, this strategy has significant risks. Because Marines often fight in the seams between the Army, Navy and Air Force (such as in the littorals), interoperability requirements are critical for the Corps. But the Corps may not completely understand the complexities of interoperability between the systems. If the Corps can't clearly articulate its Interoperability Exchange Requirements (IER's), then the lead service can't implement them. The linkage between the lead service and the Corps may not be fully defined, and a liaison may not be assigned. If a liaison is assigned and the Corps adequately defines its requirements, the lead service may not meet the Marine-specific requirement without offsetting funding. Finally, if a systems development effort is later cancelled, there is rarely a backup plan for a replacement system. The end result is you get what you pay for.

c. Stovepiped and Overlapping Development Efforts

The Corps has several “bright idea” clearinghouses. In the C2 area, there is the Marine Corps Combat Development Command (MCCDC), the Marine Corps Systems Command (MARCORSYSCOM), the Marine Corps Tactical Systems Support Activity (MCTSSA), and the Marine Corps Warfighting Lab (MCWL). Each of these agencies has a charter that gives it some piece of the automation pie. There are other non-Marine supporting agencies such as SPAWAR and contractors that also produce C2 hardware and software. Among even this short list, often items are developed without consultation or in competition with other agencies. A real-world example is the development of FOFAC/TLDHS/UCATS/MELIOS. Each of these acronyms represents four different takes by four different agencies, representing different interest groups, on the same concept of a universal spotter device. The same situation exists for C2 systems, but because of the broader span of control of C2 systems, they are often supported by the different warfighting communities. Coordination between the various systems ought to be accomplished via the Systems Architecture, but the Systems Architecture enforcement mechanisms are focused on the acquirers, not on the requirements developers. And the Systems Architecture isn't jealously protected like programs are.

d. USMC Seen as a Bit Player Among Competing Interests

The joint arena may decide that Marine Corps requirements are of lower priority than other competing interests. The other services have specific goals, and these goals may be in conflict with Marine goals. Although the effects of Goldwater-Nichols have begun to percolate through the DoD culture, there may still be some interservice rivalry that results in serious disagreements about requirements and interoperability. Most often, these disagreements are philosophical rather than technical or substantive. One example explored more fully later is the inability of the services to agree on a single common identifier for a unit.

e. Software Project Management Challenges

Software intensive systems are much harder to manage than other projects. Software is “computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.” [IEEE90, 184] The nature of software is different than that of hardware. One cannot see, touch, taste, smell, or hear software, only its effects. Software isn’t bought in traditional units (I’d like five lines of code, please), and having more software doesn’t necessarily mean more capability – in fact, it often means having more failures. Software design and coding is typically extremely error prone, and no one expects reliable software after the first attempt. In fact, the first attempt is called “alpha,” and the second, “Beta.” Often only on the third attempt is software even released.

Managing Software projects is complicated by the lack of good requirements. Few users know what they want until they see it. Customers may not be aware of what the software can do, and therefore may not ask for capability they can’t imagine. In a traditional management paradigm, by the time the customer sees software, the money has already been spent and the product has already been developed. Changes are too late. This is complicated by the DoD acquisition process, which requires people to know what they are buying before funds are allocated – almost an impossibility with software. Finally, computer power and processing ability keeps growing exponentially. Users “in the know” change requirements during software development process with the

expectation that all the interesting features they've seen advertised are available for them on their project, now. Yet changing requirements in the middle of a design effort is an indicator of a failing project.

There are no good metrics that correlate to good software. As already mentioned, size (lines of code or number of modules) tells one little about operational effectiveness. Complicated code is not necessarily good code, and one can't tell which is which without extensive experience. Military software is fundamentally different than popular civilian software in several ways. People depend on weapon system and C2 software to work correctly all the time – it is mission and time-critical. Civilian software is not so judged.

These and several other software management issues are the reason why “software is now recognized as the highest risk system component in virtually every major defense acquisition.” [GSAM00, 33] These problems are magnified across a System-of-Systems. Each system not designed, developed, and deployed to a common standard is tomorrow's legacy stovepiped system.

F. CURRENT C2 DEVELOPMENT ENVIRONMENT

1. Overview

The Marine Corps operates within the Joint C2 Development environment. The joint environment layers organizational and bureaucratic issues on top of all the other technical issues associated with C2. The Joint C2 development environment is complex, and at times chaotic. Complexities include drastic changes in DoD policy from administration to administration and year-to-year, myriad different Joint agencies with oversight, and lack of an accepted Joint Systems Architecture. The primary Joint agency is the Defense Information System Agency (DISA). This section discusses DISA and other Joint players relevant to the AFATDS/IOS discussion.

2. DISA

The Defense Information Systems Agency is a DoD support agency under the control of the Assistant Secretary of Defense for Command, Control, Communications and Intelligence [ASD (C3I)]. The Agency began in 1960 as the Defense Communications Agency (DCA) to consolidate the communications functions common to the military departments. In 1991, the name was changed to DISA. Functions given to DISA include maintaining the Global Information Grid, which is the sum total of all deployed communications and networking assets owned by the DoD. DISA has an interoperability directorate, and is required to test and certify C2 systems for interoperability according to the DII COE compliance scale, mentioned below. The Joint Interoperability Test Command (JITC) in Fort Huachuca, AZ conducts this testing. Of note, the services have been delegated this authority to certify their own systems for DII COE compliance. JITC has assumed responsibility only for truly joint systems such as GCCS. The Army certified AFATDS.

There is a joint Systems Architecture. DISA maintains the Command, Control, Communications, and Computer Information Support Plan (C4ISP). The C4ISP is a collection of documents and pictures that express the “as-is” and “to be” operational architectures in DoD. The C4ISP forms the basis of the Marine Corps’ “C4I for the Warrior” (C4IFTW) concept. However, the C4ISP has taken years to develop, and is inclusive of all services and systems. Therefore it is aimed at the lowest common denominator, which hinders interoperability.

DISA also maintains the Joint Technical Architecture (JTA). The JTA is a collection of computer and interface standards that all automated systems are supposed to meet DoD-wide. Many of these standards are the same as civilian standards. An example JTA standard is the Transmission Control Protocol/Internet Protocol (TCP/IP). The JTA forms the basis of all technical architectures. Meeting the specifications of the JTA is not terribly difficult for COTS products, since most of the standards are commercial standards to begin with. Meeting the requirements of the JTA are a little more difficult for GOTS and specialized systems (including legacy systems), as they

must be designed in and may require code changes. Yet, meeting the JTA does not guarantee interoperability.

3. ASD, AT&L (Interoperability)

The office of the Assistant Secretary of Defense, Acquisition, Technology, and Logistics (ASD, AT&L), has an Interoperability director. Projects include the “Global Information Grid” and the Joint Warfighting Capability Assessment Reports. [DATL02] The relationship between this OSD and DISA, which as previously stated, is under ASD C3I, is unclear. Trying to contact a member of their office proved fruitless. This is another example of the continuing confusion surrounding “who’s in charge” of information in the DoD.

4. GIG COE

The original name for the Global Information Grid was the Defense Information Infrastructure (DII). In the 1998 DII Master Plan, the DII was described in part as, “...the web of communications networks, computers, software, databases, applications, weapon system interfaces, data, security services, and other services that meet the information processing and transport needs of DOD users, across the range of military operations.” [DIMP98] In October 2001, the name was changed by DISA to the Global Information Grid (GIG). [FCWK01] The name change was meant to reflect a “new way of doing business,” focusing on connecting people a la the internet, vice a monolithic, single-purpose system. In fact, the term “GIG” is not often used, and most documents still refer to the DII COE. However in either case the intent is to standardize the logical underpinnings of all DoD information systems. To that end, the Common Operational Environment (COE) was established.

The roots of the COE go back to the Joint Maritime Command Information System (JMCIS). In 1995, Admiral Gauss was working on this Command and Control System for the Navy at SPAWAR. When he went to DISA to become its director, he took the Operating System and Common Application Program Interfaces (API’s) and made them the de-facto standard for the emerging Global Command and Control System

(GCCS). The collection of these API's and other standard software was termed the Unified Build (UB). ([BUDD02] and [BULL02]) The GIG COE (or DII COE) is built as shown in figure 4 below.

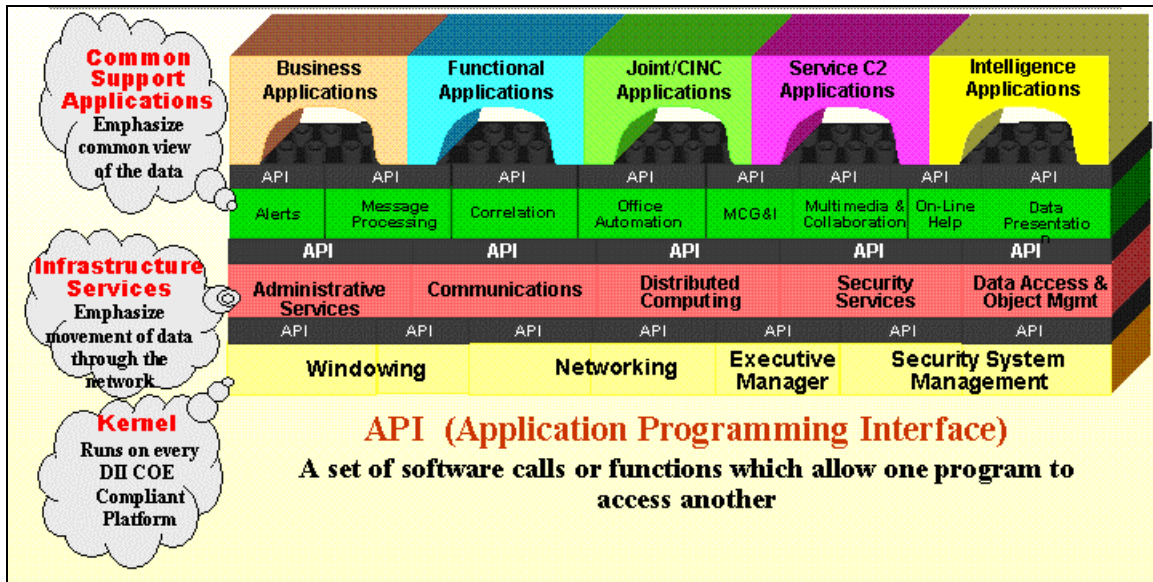


Figure 4. The DII COE Conceptual Model (After [WALK01, 3]).

The model has four layers. The first three are listed in the bubbles on the left of the diagram. The bottom layer is the Kernel, and interacts directly with the Operating System. The second layer up is “Infrastructure Services,” and contains basic communications and administrative functionality. The third layer is the common support applications. It is this layer that individual Command and Control Systems were duplicating again and again – now if someone builds to the COE, those functions do not need to be re-written. The fourth and final layer contains the business and functional applications, shown as arches built on the framework of the first three layers.

Eight Levels of DII COE compliance are specified in the Integration and Run-Time Specification. (DII COE I&RTS). Now that DISA was mandating use of the DII COE, C2 system program managers and hardware vendors strove to get “DII COE compliant,” so they could be certified. Of course, compliance with the DII COE standard will facilitate interoperability. DII COE compliance is necessary but not sufficient to ensure interoperability. The only problem was that all these programs have long lead times, different leadership, and institutional goals that generally supercede

interoperability goals. In order to ensure everyone could be certified, eight levels of DII COE compliance were introduced. The 8 levels are shown in the Table below. Notice that system-of-systems interoperability is not addressed until level 7. Thus, it is not good enough to have a DII COE (or GIG COE) compliant system, but it must be at least level 7 or 8 compliant to be truly interoperable. This is a source of confusion for many warfighters.

Level	Name
1	Standards Compliance
2	Network Compliance
3	Platform Compliance
4	Bootstrap Compliance
5	Minimal DII Compliance
6	Intermediate DII Compliance
7	Interoperable Compliance
8	Full DII Compliance

Table 2. DII COE Levels of Compliance (After [DIRS97].)

5. JROC

The Goldwater-Nichols Act of 1986 reformed the Department of Defense with the purpose of reducing the excessive influence of the four services and increasing the power of the Joint Chiefs of Staff to meet the country's defense needs. One particularly troubling area of interservice rivalry was in acquisition and procurement of non-interoperable materiel. This resulted in an inability to fight together because systems were not interoperable. It also wasted resources, as the services would buy duplicative items. [LOCH02, 437] Therefore, the act strengthened the role of the Chairman of the Joint Chiefs of Staff (CJCS) and gave him a Vice Chairman (VCJCS). The VCJCS chairs the Joint Requirements Oversight Counsel (JROC), which is composed of the Vice chiefs of each of the military services.

The Joint Requirements Oversight Counsel (JROC) is the senior military body charged with reviewing all major defense acquisition programs for alignment with Joint warfighting goals expressed in the Universal Joint Task List (UJTL). The JROC validates new mission needs, reviews program alternatives, and evaluates programs on the merits of their Key Performance Parameters (KPP's) along the life of the program. If

a program is not meeting its KPP's, the JROC can recommend that the program be modified or cancelled. Systems interoperability is a KPP. By controlling the purse strings via joint oversight, the intent was to force the services into interoperability. The JROC could be the one single unifying entity in the search for interoperability, as they have the military rank and the statutory authority to make difficult calls on funding. The JROC has the responsibility and authority to require interoperability fixes to programs.

Yet in the fifteen years since Goldwater-Nichols, true joint acquisition reform has not occurred. Instead, the JROC operates by consensus, which means if a service wants their project to go forward, the other vice chiefs agree to let it go forward if theirs will too. [LOCH, 443] Several previous JROC chairmen have expressed frustration that deep-seated service resistance thwarts their best intentions. [LEDE99, 95] Further, congress has never called the Chairman of the JROC to testify on budgetary issues, despite its statutory role. [LEDE99, 95] This lack of credibility with congress translates to a lack of ability to meet oversight goals. The one internal oversight body with the power to force interoperability has not lived up to expectations.

6. SPAWAR Charleston

The Space and Naval Warfare Systems Center, Charleston, South Carolina, serves as the project engineer for the Marine Corps on several major Command and Control Systems. The table below lists several of the systems that SPAWAR Charleston currently manages for the Corps.

Acronym	Short Title	Description
GCCS	Global Command and Control System	Provides the fielding plan, hardware, and software builds for the GCCS terminals in the Marine Corps
DACT	Data Automated Communications Terminal	Provides the terminal for Forward Observers to enter data for transmission to C2 systems such as AFATDS
EPLRS	Enhanced Position-Location Reporting System	Provides digital radio communications between C2 nodes.
IAS	Intelligence Analysis System	Collates and provides intelligence information from multiple intelligence sources. IAS software is what differentiates IOS (V2) from IOS (V1).
IOS	Intelligence-Operations Server	DII COE compliant hardware and software. IOS (V1) is TCO software; IOS (V2) is TCO and IAS software.
TCO	Tactical Combat Operations	Software that maintains the Common Operational Picture at the Division and below.

Table 3. Pertinent C2 Projects at SPAWAR Charleston

From an interoperability standpoint, it is advantageous to have the several system developers resident in the same building. As discussed later with IOS, there has already been significant consolidation among the various systems and funding lines. Further, SPAWAR Charleston clearly has support of the warfighter in mind. Specifically, they have created a communications website, using collaborative technologies, that allows all users with an HTML browser and appropriate permission to communicate with Project Officers, Requirements Generators, and project engineers. [SPAW02] Each project has Point of Contact listings and a “discussion room” that permits valuable group interaction. This website, called TACMOBILE, is one of the critical items in making C2 systems usable in the fleet.

THIS PAGE INTENTIONALLY LEFT BLANK

III. THE UNIFIED MODELING LANGUAGE

A. INTRODUCTION

Software must model the real world. The fidelity of the software model to the real world is a good indicator of whether the software product is effective or not. An example from the financial world could be a bank account. In the past, there was a stack of money that customers gave to the bank. The customer had physical items (gold or bills and coin) to hand over to the bank, that the bank kept. The bank may have paid interest, which could be withdrawn along with the principal. Either way, the customer knew where the money was located, and could make a withdrawal. Then along came software and automated record keeping. There may not be any bills or coin in the bank vault, but the customer expects the software to accurately model his or her bank account, including the interest. Both the bank and the customers trust the software with a very important asset – money. Further, the customer can now do so much more – say go to Europe and withdraw money denominated in Euro’s from his or her same account.

The Unified Modeling Language (UML) has become the de-facto standard software modeling language. The UML is useful for eliciting the business rules and processes that form the basis for software modeling, for generating requirements, for design, and for coding of object-oriented systems. The UML provides an easily extensible common visual language for representing objects and their interactions. Grady Booch, one of the principal authors of the UML, states; “UML is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system.” [BOOC97, xv]

UML is an effective solution to eliminate the Tower of Babel problems previously extant in the Software domain. Previous methods of software modeling presentation were heavily personality dependent, and the semantics and syntax of the various modeling languages were a constant source of debate. UML standardizes the language semantics and syntax, while providing a method to easily extend UML to fit those situations where the basic language can’t adequately communicate.

1. Object Orientation

The portion of the world that a particular software application attempts to model is known as the problem domain. Objects are things, concepts, or entities associated with the problem domain. [LARM98, 6] Example objects in the banking problem domain could be money, ATM, and account. Example objects in the C2 problem domain could be friendly unit, country boundary, and map. Object-oriented analysis (OOA) focuses on identifying all the relevant objects in the problem domain that allows answering the questions the software was designed for. In the banking example, if we only want to know about savings, we would model savings accounts and a check might not be an object. This is a model with low fidelity. If we wanted to have a banking model with higher fidelity, then we could model savings, checking, and loan accounts.

Objects in OOA are associated with other objects. The job of the analyst is to capture the attributes of the objects, the associations the objects have with other objects, and the actions the objects can take. This data is collected along with the desires of the customer about what the software system is supposed to do. These requirements and object descriptions lead to a more formal definition of the requirements for the software system. How the objects interact with each other is captured not only in the associations, but also in the business rules that are in the problem domain. These business rules are captured in a “Business Use Case.” The Business Use Case is a step-by-step description of what happens in a given situation.

Object-oriented analysis leads to object-oriented design. Object-Oriented Design (OOD) focuses on designing software objects that correspond to the objects discovered in OOA. The design is then encoded into an object-oriented programming language. During the design stage, objects are given attributes and methods. An attribute is a particular feature of the object, while a method is what the object can do. Returning to the banking example, an account has an attribute called “balance,” which is expressed in dollars and cents. If we imagine an account can check its own balance and report it, then account could have a method called “get balance” (or getBalance) which would report the balance in dollars and cents. Following the same concept in the C2 world, a friendly unit

could have an attribute indicating how large the unit was (numbers of people), and it could change geographic locations (move).

OOD leads to object oriented code. In order to do this, the essential features of each set of the same or similar objects are abstracted into object *Classes*. A *Class* is a framework for objects of the same name. Returning again to the Command and Control example, a Class might be “Friendly Unit,” with objects in that class being 1st Platoon, 2nd Platoon, and 3rd Platoon. These platoons are *instances* of the Friendly Unit Class. Once they are created (creation is known as instantiation), they keep all their own attributes and methods until specifically destroyed. If the Friendly Unit Class has a *Move* method, then each platoon can move independently and remember where it is.

Object orientation is just one way to look at the world, but object-orientation captures data and actions in a way that supports some basic software goals. The primary goal is working software with less complexity for the developers and programmers. Object orientation supports that by hiding the internals of objects from other objects so they can’t interfere with each other. It also supports software reuse by allowing objects to inherit attributes and methods from other objects. Finally, objects can remember things about themselves, an important attribute for software systems.

2. Advantages of UML

There are many ways of modeling real-world processes and converting those models into working software. In fact, there are too many methods. The primary advantage of UML is that it has gained wide acceptance in the software development world and has become the language of choice to represent OOA/OOD. Of course, it became popular because UML had some specific advantages over previous methods.

UML is visual, allowing people to visualize how software models interact. The visual components are standardized and fairly easy to learn and manipulate. UML is extensible. This means that there is a standard way of adding features to the language if there are items in the problem domain that do not fit the standard. Being object-oriented, UML models lend themselves to direct conversion into code using an object-oriented language such as Ada, C++, Java, or SmallTalk. This conversion can be done

mechanically, and in fact there are several software packages that will turn sufficiently detailed UML models into computer code.

3. Why UML is Useful for this Project

UML was developed starting in 1994. [BOOC99, xix] The AFATDS project started in 1981, and used a different software modeling process to describe the business rules in the Fire Support Problem domain. The primary precursor to the IOS software suite was being developed from other software in 1995, and also did not use UML. Yet, UML allows presentation of complex topics in a visual, simple way. Using UML permits an appropriate level of abstraction. Finally, since both systems were developed completely differently, UML serves as a common description language of the military's business rules and these systems.

B. USE CASES

The first step in object-oriented analysis of a problem domain is to find objects and business rules. Objects become classes. Business rules become Use Cases. Decisions are made about what the software we are designing is supposed to do. In order to better understand the UML, we will model a simple process from the Command and Control Arena – create and move an infantry platoon. The name of the platoon is 1st platoon, Alpha Company, 1st Battalion 2nd Marines. It is composed of 3 squads. It is full strength. It is at a particular location, location X. The Commander orders it to move to location Y. The software system we are designing takes automatic data updates over tactical radios from a GPS receiver at the platoon. The Tables below lists the requirements for the software System and the Use Case for this scenario.

User Requirement:	Sub-Category	Requirement Number
Store Unit info	<ul style="list-style-type: none"> • Store Name • Store Unit Strength 	R1.1 R1.2
Track Unit	<ul style="list-style-type: none"> • Automatically Update location 	R2

Table 4. Unit Tracking System Requirements

USE CASE:	Unit Tracking System: Track Unit	
Actors:	Commander, Friendly Unit	
Purpose:	Accurately model a moving unit.	
Overview:	This Use Case describes a command and control system that tracks moving units using automatically updating GPS coordinates. The system must be powered on and communications must be set up prior to entering this Use Case.	
Type:	Primary and Essential	
Cross References:	R1.1-1.2, R2	
Actor Action		System Response
1. This Use Case is initiated when an operator (working for a commander) creates a unit in the system. The operator enters the Unit Name, Unit Strength, and Unit communications ID.		
		2. The system loads the data and listens for messages from that unit, using the Unit Communications ID.
3. The unit GPS receiver periodically reports its position.		
		4. The system displays the unit as an icon on a map displayed on the screen. Information includes the Unit Strength and Unit Name.
5. The commander orders the unit to move.		
		6. The system displays the unit as it moves.
ALTERNATE Courses: Step 4: If the system does not detect the unit communications ID within a commander-definable period, the system alerts the operator that communications have not been established.		

Table 5. Unit Tracking System Use Case

1. Actors

Actors are people that use the software. Actors can also be other software that uses the software we're designing. The key is that Actors are outside the software we are designing. In table 5 above, the Actors are the Commander and the Friendly Unit. The Friendly Unit contains a sensor. The sensor is not a person but a device that sends its position. The sensor could be in this Use Case if the designers thought it would be important to accurately model the situation. Also notice that "Friendly Unit" is an actor, not 1st Platoon. 1st Platoon could be used, but the system is designed to handle many reporting units besides 1st Platoon. Finally, often positions that are not critical to the

understanding of the Use Case are not listed. This is the case with the operator, who needs to be there, but acts as an interface between the Commander and the system.

In UML, a Use Case is diagrammed as shown in figure 5 below. The bubble is the Use Case, while the box represents the system boundary and the stick figures represent the Actors. There can be several Use Cases inside the system, and there would be a Use Case narrative for them as well. The lines that connect the Actors to the Use Case indicate an association between the Use Case and the Actors. The nature of the association is described in the Use Case narrative.

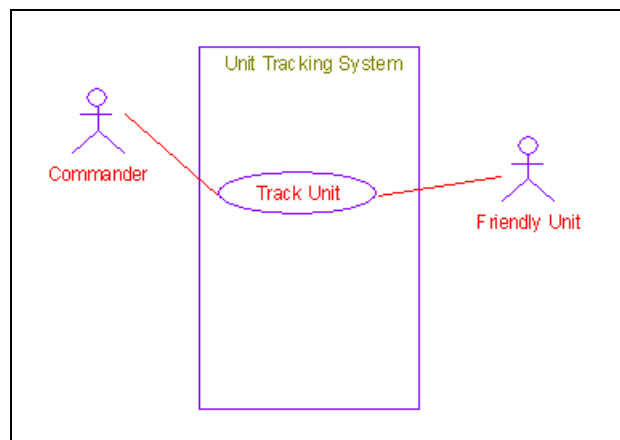


Figure 5. Unit Tracking System

2. Classes

Continuing with this example, two classes that can be derived from this Use case are the Friendly Unit Class and Map Class. The Friendly Unit has attributes of Unit Strength, Unit Name, location, and Communications ID. The Map Class has lower left hand corner location and scale. There are many other attributes that can be attached to these classes. There is no limit to attributes that can be associated with Classes in the model, although of course there are practical limits to how much a computer system can store and process. The Friendly Unit has a method called Move, which takes a location and moves the unit to that location. In the real world it matters how location is represented – let's say for this example that locations are in the Military Grid Reference System (MGRS), and that six digits are sufficient. When designing an actual system, all these types of requirements (what type of data and how its represented) are critical for the

system users to define for the system designers. Figure 6 below shows how these classes are represented in the UML.

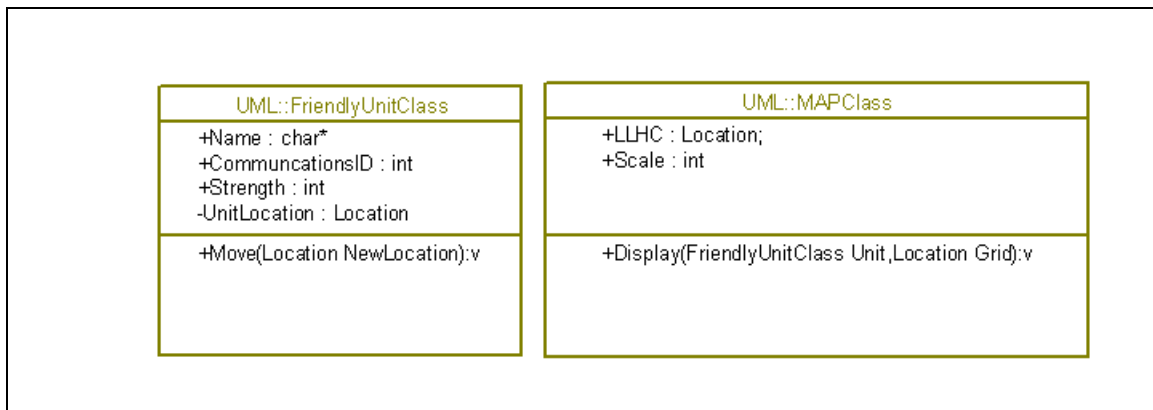


Figure 6. UML Example Class Diagram

For each class, the top part of the class box shows the Class Name. The middle part of the box lists all the Class's attributes. The bottom box lists the Class's methods. The plus sign in front of each attribute and method indicates that the attributes and methods are visible outside the class. The minus sign in front of UnitLocation in the FriendlyUnitClass means that attribute is not visible outside the class. What this means is that a new location can only be accessed via the Move method. This is an example of data hiding – now the location of the unit is protected from some programmer errors. Notice also that each attribute is defined in terms of the language – in this case, Strength is an integer variable, and it is up to the system designers to define (in the comments and by proper usage) that Strength means number of people. Or an alternate tack can be taken where the designers define their own data type. An example user-defined type is Location in the figure above. Another Class diagram would show what a Location consists of.

C. COLLABORATIONS

After the Classes and Use Cases have been determined, the Use Cases are evaluated to determine the order of activity in the diagrams. There are two major methods used to show collaboration in the UML. The first method is with a collaboration diagram, and the second is with a Sequence Diagram. The two diagrams are functionally very similar. Both diagrams are shown here for a portion of our example Use Case.

Figure 7 shows a collaboration diagram for the action of the Commander moving First Platoon. In this example, the commander orders the unit to move. This first action is reflected by First Platoon's sensor periodically reporting a new location, which is displayed on the current map. Notice the associations show the use of the methods previously declared for the various Classes. Also notice that 1st Plt is an instance of the Friendly Unit Class. These actions are not constrained by time. The only thing specified is the order of execution.

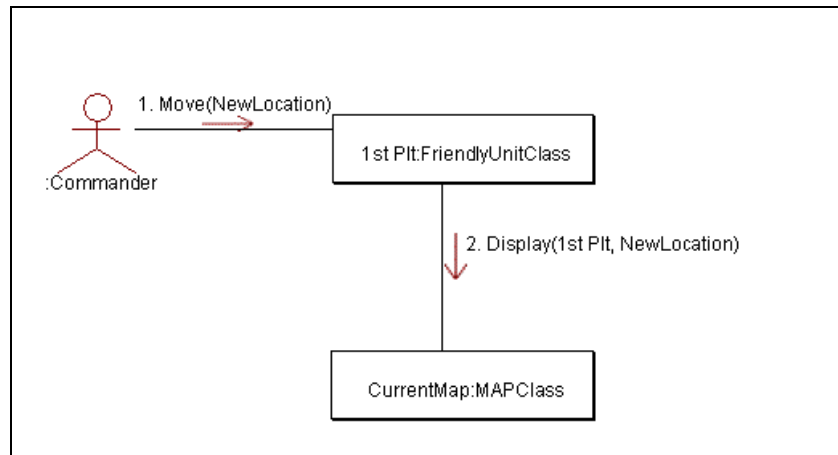


Figure 7. UML Collaboration Diagram

Figure 8 shows the equivalent UML Sequence Diagram. In this diagram, the Objects and Actors are listed across the top. Each object and Actor has a lifeline corresponding to it. Time flows from top to bottom. Action is signified by sending messages. On the diagram, messages have an arrow going from the originator to the receiver. Exactly what time something happens is not specified, but again the order of message delivery is specified.

The boxed message in the middle of the diagram is a comment, and is placed over the system boundary. Notice the Commander is external to the system. Of course, First Platoon is outside the system as well, but the actual first platoon is not on this diagram – only the 1st Plt object inside the system is depicted. There can also be lines going from an object back to the same object. This represents computer processing internal to an object. Often, these lines indicate a process that takes extra time or shows processing that supports further message passing in later steps.

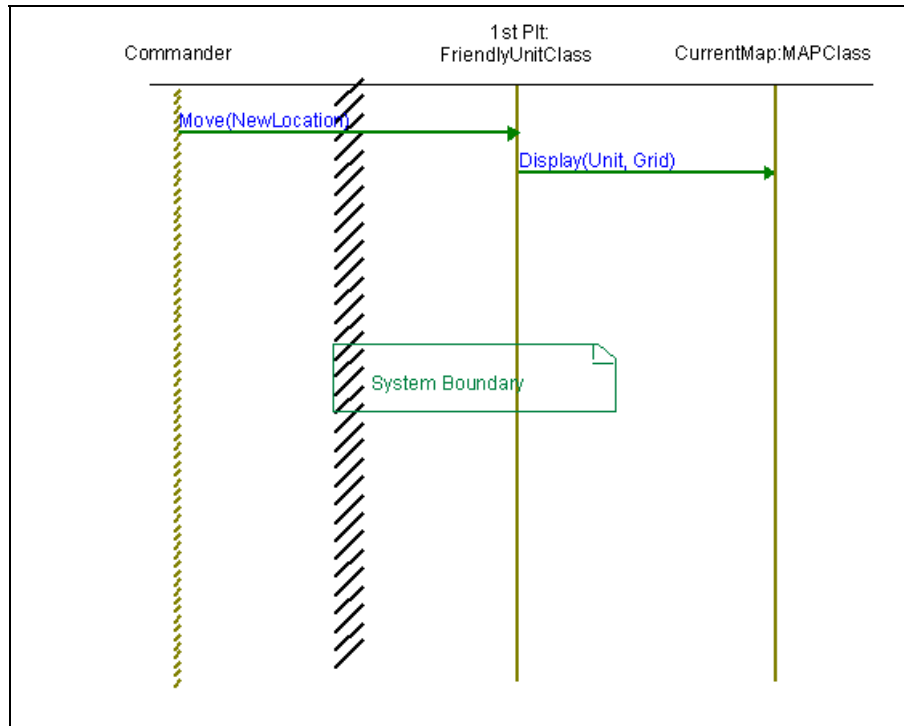


Figure 8. Example Sequence Diagram

D. CONCLUSION

All the drawings and documents created during the software development process are called artifacts. One software development process (the Unified Process) has broken down the phases of software development into Inception, Elaboration, Construction, and Transition. The diagrams produced here cover parts of the inception and elaboration phases. However, in order to develop software, several other artifacts are needed, at a greater level of detail. These artifacts can also be expressed in UML and its associated tools. The artifacts shown above are sufficient to complete the analysis needed for this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. THE INTELLIGENCE – OPERATIONS SERVER (IOS) SOFTWARE SUITE

IOS is the Marine Corps program that supplies the intelligence, communications and processing capability needed to meet the Commander's requirement for C2 at the tactical level. IOS receives, fuses, displays, and disseminates selected operational input from the MAGTF's other C2 systems. It is intended to meet the need for the commander to conduct all aspects of C2.

A. HISTORY

1. Introduction

The history of the Intelligence-Operations Server is not well documented. Unlike AFATDS, which has had the sponsorship of the Army, the artillery community, a program office and program managers for the length of the program, IOS evolved from a collection of different program offices and programs that were developed over a period of years. In general, the software from which IOS is composed was written to meet the needs of the Navy and the Maritime C2 environment. This environment and the Navy's unique approach to Command and Control drove many of the software design decisions. Interoperability with ground C2 systems was not a primary goal of the original systems. In order to better understand the drivers behind IOS, it is appropriate to study the aspects of the Maritime Command and Control Environment.

2. The Maritime Command and Control Environment

Each Naval ship has a significant amount of autonomy. Likewise, a Naval battlegroup also has significant authority and responsibility to carry out assigned tasks with limited involvement from higher units. Area coordination between battlegroups is accomplished through assignment of geographic Areas of Responsibility (AOR's). AOR's are large, allowing for significant maneuver room for the individual battlegroup. When at sea, communications between ships have always been severely limited by

distance and the mobile nature of seagoing vessels. Therefore, ships tended to be self-contained, self-supporting units with strong internal bonds and the ability to carry out assigned missions with a maximum of autonomy.

In this environment, the focus on C2 functions such as intelligence gathering was on internal needs, rather than on reporting to higher units or maneuvering as part of a group. Typical questions were: how far away is the nearest land? Where is the rest of the battlegroup? How far is the enemy from me and how fast is he closing? Once the captain of the ship gained enough situational awareness, he could make decisions and act autonomously. Radar, observer, and other sensor feeds were processed and collated aboard ship in the Combat Information Center (CIC). Many times, decisions could be made with a minimum of information – for example, a Radar contact could be deemed hostile if it was moving toward the ship at a high rate of speed without radio contact.

As radio communications became more robust, there was a natural desire to share information between ships in a battlegroup in order to better task organize. For instance, in a Carrier Battlegroup, one ship would be given the Anti-Air Defense Coordinator (AADC) role, responsible for protecting the battlegroup from air attack. This meant that such a ship would need the complete air picture for some distance around the battlegroup in order to protect it. Because of the limited communications bandwidth and the multicast nature of radio communications, radio networks were designed to pass the minimum amount of information possible. Only the most important information was passed. This information amounted to the type of contact, and its location, course, and speed. This position-location information was called a “track” because in general, the information was displayed as an icon with a leader indicating course and speed on a display. Sailors initially maintained these Manual C2 displays by writing the information with grease pencils on clear boards. As these manual displays were automated, the terminology remained.

In naval circles, a track is a single item to be displayed in the COP, like a ship. Many details about the ship were unimportant for immediate decision-making. Important information about a track included its location, bearing, and speed. It also mattered what type track it was - whether it was above, on, or under the sea. As Naval C2 systems

developed into multi-service systems, the definition of a track began to change. In 2002, the Joint Chiefs of Staff defined a track thus:

A track is a single entity reported on the COP such as an aircraft, ship, TBM [Tactical Ballistic Missile] or emitter location. A track can also designate an aggregation of military personnel, weapon systems, vehicles, and support elements or any other operationally significant item. [JCOP02A]

One can see the naval history to this term in the first sentence – a track was traditionally a single entity reported to the ship. The second sentence demonstrates the change in the definition of the term beyond its original meaning. However, changing the definition of a track in a dictionary to include ground scenarios is much different than independently developing a concept of the data needed to represent a ground unit.

In an ocean environment, maps could be rudimentary – the ocean is flat. Navigators handled navigation with specialized maps and systems, and generally navigation was done without the aid of C2 systems. Important spatial information for the C2 systems were the AOR and other man-made control measures, and the location of the shoreline in relation to the ship. It was in this context that Naval Command and Control Systems were developed.

3. Joint Maritime Command Information System (JMCIS)

Although the history of Naval Command and Control systems starts long before JMCIS, JMCIS is the father of the current Naval C2 systems, to include the Global Command and Control System – Maritime (GCCS-M). In fact, DISA took the Unified Build from the JMCIS program. Figure 9 below shows the evolutionary development of JMCIS from the myriad stovepiped Naval Command and Control Systems developed during the 1970's through 1995.

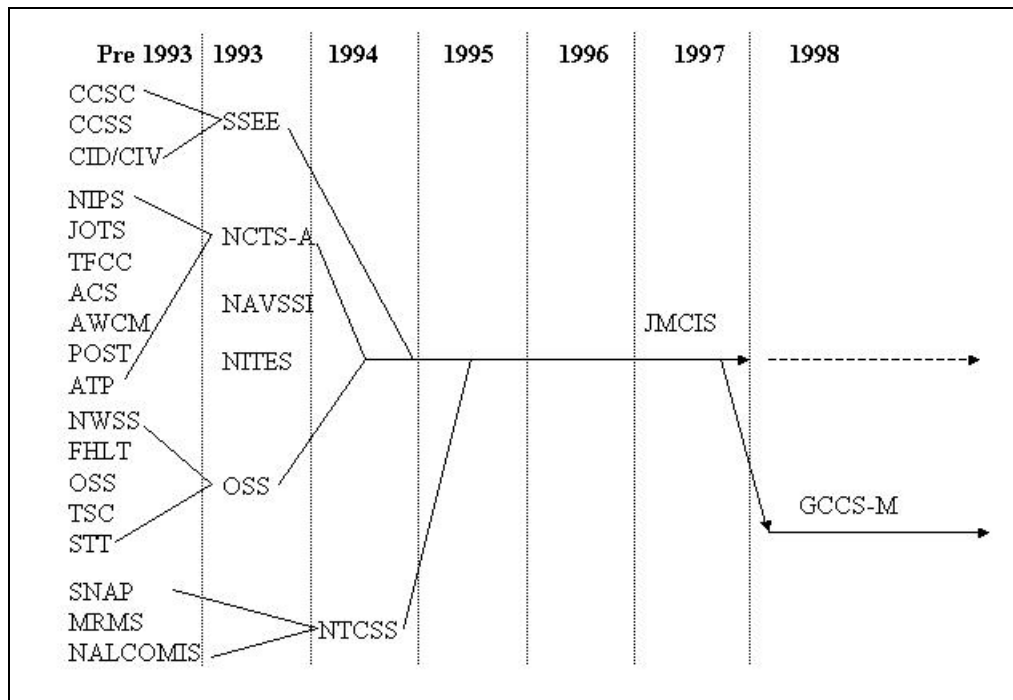


Figure 9. JMCIS Lineage (After [BUDD02])

The Space and Naval Warfare Command (SPAWAR) had responsibility for developing JMCIS. Understanding exactly what each acronym stands for in the diagram is unimportant. What is interesting about the lineage shown in Figure 9 is its evolutionary nature and the ability of the Navy to get the developers of the other C2 systems to relinquish control of their Programs of Record and budget in favor of a common system. Of course, many technical and systemic hurdles had to be overcome in order to completely integrate these programs. Rather than integrating systems, JMCIS engulfed these systems – taking the funding and personnel from each.

From an organizational standpoint, integration became feasible as people realized the massive duplication of effort inherent in the development of different systems. Each system had up to 80% of its software functionality in common with the others. For example, each system had methods to display charts, to place tracks on the chart, to send and process alerts, to “chat” (collaborate between nodes) and send messages, etc. Some programs were even sharing code for these common functions. [BUDD02] Using the JMCIS program, SPAWAR separated these common functions into the “Unified Build (UB).” The UB packaged these common modules into a “Government Off the Shelf

(GOTS)” package with common Application Program Interfaces (API’s) for each segment³. These API’s were standardized methods for programmers to access the functionality contained in the UB. The net effect was for a given specialized C2 system, the amount of work needed to deliver the required functions drastically decreased. To accomplish a domain-specific mission, each developer had only to develop a segment that then called the UB API’s for services. DISA took the UB to form the basis of the Common Operational Environment (version 3.x). The UB also became the core system for the Global Command and Control System – Maritime (GCCS-M).

JMCIS and the follow-on systems were appropriated and modified because they were successful Naval C2 systems. However JMCIS’ development was not well documented, and the requirements and design decisions surrounding JMCIS have never been put to paper or studied. As evidence of this lack of developmental rigor, there is still no delineation in the literature between GCCS-M and JMCIS. This is made clear in the current (version 3.1.2.1) GCCS-M Segment Description Document, which has a disclaimer that JMCIS terms may be seen throughout. [GSDD01] This lack of any sort of documentation, especially of requirements and design decisions, hinders development of interoperable systems.

4. Tactical Combat Operations (TCO)

In 1992, the Marine Corps Combat Development Command (the USMC organization responsible for operational requirements) issued a Mission Need Statement (MNS) for a ground C2 system. Specific needs were “...to receive, fuse, display, and disseminate selected operational input from the MAGTF’s other C2 systems.” [TMNS92, 1] The term “operational input” is not defined in the MNS, but it can be surmised from context that the authors meant electronic data. Also inherent in this need is the requirement for interoperability with the MAGTF’s other C2 systems. The authors

³ In JMCIS, GCCS, and DISA terminology, a “segment” is a particular function seen by the user. A segment often has more than one software module, but provides a seamless interface to the user. For example, the Joint Mapping Toolkit (JMTK) segment has a client software module and a server software module, but the user only interfaces with the client module. [GSDD01]

further emphasized interoperability in the MNS by explicitly listing many current C2 initiatives that TCO had to interface with. Yet, many of these systems were “currently in development” when the MNS was written. In the fire support area for example, an interface was required with “Fireflex,” a system the Corps was to develop on its own. [TMNS, 5] Fireflex was never developed due to lack of funding and the ultimate choice of AFATDS as the Fire Support System of Record.

By 1995, the Marine Corps Combat Development Command had developed a “Concept of Employment” (COE) for TCO that applied the TCO to the MEF down to the Battalion and Squadron levels. [TCOE95] This tactical level of warfare was not well served by the developing Global Command and Control System (GCCS), which was more suited to the strategic and operational levels. By differentiating itself from GCCS, TCO became a Program of Record and had its own funding and project office. Also, GCCS was a joint project, and the Corps wanted control of this tactical system. The COE again emphasized integration of TCO with other MAGTF C2 systems from the functional areas of maneuver, fire support, intelligence, air operations, combat service support, and Command and Control Warfare. By this time, the Operational Requirements Document (ORD, generated by the Marine Corps Systems Command as an acquisition document) listed a requirement for TCO to interface with AFATDS, with the interoperability standard being reached by 3rd Quarter FY 1997. [TORD95, 7]. However, what “interoperability” meant was not defined in any TCO document. In September 1995, the TCO Integrated Program Summary had officially tied TCO to the JMCIS project. [TIPS95]

From an operational standpoint, the connection between JMCIS and TCO is not immediately apparent. TCO was supposed to meet the Ground Commander’s need for a C2 system. JMCIS is a maritime system. These different environments have markedly different requirements. Yet, there were several advantages. Parallel development allowed the Marine Corps to save money when compared to an independent development effort. Also, it was clear the Navy was serious about their own system integration in JMCIS. JMCIS software was showing promise. This gave Marine leaders confidence that the JMCIS project was viable. Finally, this was about the time that DISA began mandating a Common Operating Environment, and so TCO was ahead of the game in

meeting joint requirements. TCO as deployed therefore became a subset of JMCIS segments deployed on Common Hardware.

5. Intelligence-Operations Server (IOS)

The desire to integrate disparate systems continues. Even the use of language is important in this initiative. The term “Intelligence-Operations Server” is descriptive of where the Corps wants to go – one server at each maneuver unit that will “seamlessly” support the commander’s information requirements and C2. In order to achieve this goal, the intelligence functions contained in other programs such as the Intelligence Analysis System (IAS) must be merged (on the same server) with TCO. Maneuver units do not have a plethora of people to maintain systems, and having one server instead of two is an obvious benefit. Also, the Marine Corps wants standardized hardware across the FMF. See figure 10 below for an idea of the size of the current IOS hardware suite – 4 person lift. The box contains the Sun Netra server.



Figure 10. IOS in a tactical environment.

In 2001, TCO became IOS (version 1). There is no Mission Needs Statement (MNS) or Operational Requirements Documents (ORD) for the IOS. The Program

Manager instead uses the TCO Program of Record, ORD, and funding. [PECK02] The Corps still receives funding for TCO, but it is spent on IOS. IOS (version 2) is TCO with the addition of IAS. SPAWAR Charleston is the system developer for TCO, IOS, IAS, and eleven other systems for the Marine Corps. Because the same center develops these ground products, there is significant synergy between the different programs. The IOS (Version 1 and 2) software runs on a Sun Netra 1125t Station. The operating system is Solaris V. 2.5.1. There are no current plans for IOS to migrate away from Solaris.

B. OPERATIONAL REQUIREMENTS

This discussion focuses on IOS (v. 1) as fielded. The process used for requirements will be to list a subset of the user requirements derived from the U.S. Marine TCO ORD [TORD95] and TCO COE [TCOE95]. There are several problems with this approach. First, the authors of the ORD and COE used the English language to describe C2 concepts that are not easily described in words, while attempting to minimize the respective documents. They were unable to accurately state what the system should do. Few example C2 systems existed in 1995 for comparison. Second, C2 requirements have evolved significantly since then, and there is no documentation extant today that accurately lists current requirements. As with most computing systems, the user needs to see a prototype before “it feels right.” When it comes to generating requirements, users often say “they’ll know it when they see it.”

There is no history of program development. Therefore, one must use engineering judgment to interpret what functionality is in the current software and extrapolate back to the requirement that drove that functionality. From the derived operational requirements, a UML model with Actors, Use Cases and Class diagrams will be produced. Operational requirements are better understood in the context of the Marine Corps tactical organization for combat. For readers unfamiliar with U.S. Marine Corps Organization, Appendix B clarifies U.S. Marine combat and fire support relationships and lists one scenario where IOS and AFATDS is used.

1. Selected User Requirements

The Requirements listed below are listed in a building-block fashion. The primary requirement for a Common Operational Picture (vice just an Operational Picture) is communication between nodes, so it is listed first.

User Requirement:	Sub-Category	Requirement Number
Communicate Digitally	<ul style="list-style-type: none"> Establish Communications Manage Alerts Autoforward messages Filter incoming/outgoing messages Unicast data Broadcast data Manage Newsgroups 	R1.1 R1.2 R1.3 R1.4 R1.5 R1.6 R1.7
Manage Text and Graphics	<ul style="list-style-type: none"> OPLANS Orders Messages Reports Presentations 	R2.1 R2.2 R2.3 R2.4 R2.5
Manage Maps	<ul style="list-style-type: none"> Display DMA products Change scale 	R3.1 R3.2
Manage Friendly Tracks	<ul style="list-style-type: none"> Update Locations Manage Track Data Correlate and aggregate Tracks Predict Future Locations (Routes) 	R4.1 R4.2 R4.3 R4.4
Manage Enemy Tracks	<ul style="list-style-type: none"> Update Locations Manage Track Data Correlate and merge Tracks Predict Future Locations (Routes) 	R5.1 R5.2 R5.3 R5.4
Manage Overlays	<ul style="list-style-type: none"> Add/Delete/update boundaries and graphics Associate Overlays with OPLANS Activate/Deactivate Overlay Filter Units based on Overlay Send Operator Alerts based on Unit/Overlay Interaction 	R6.1 R6.2 R6.3 R6.4 R6.5

Table 6. Selected IOS User Requirements (After [TORD95], [TCOE95])

2. Actors

As explained in Chapter III, in the UML, actors are the people, things, or systems that interact with Use Cases. They are external to the IOS system. In the military, and thus in IOS, the same sequence of Actors is repeated for each level of the military hierarchy. For example, there is a Battalion Commander and a Regimental Commander. These are specializations of the Actor Role of “Commander.” These specializations are reflected in the IOS when it is important for processing. The following table lists the characteristics of the principal Actors in the IOS.

Actor Role Name	Description	Example Instances
Commander	The commander role is that of the person given responsibility and authority to prosecute a campaign. The commander sets warfighting policies that are then encoded into IOS (an example encoding is overlays). The commander expects the information and data in IOS to be correct, and looks to IOS for situational awareness.	<ul style="list-style-type: none">• Battalion CO• Regimental CO• Division CO
Staff	A member of the commander’s staff fills the staff role. The staff is responsible for implementing the Commander’s policy. Staff provides human oversight of IOS operations. At least one member of the Staff serves as the COP Track Correlator (The “TOP COP.”)	<ul style="list-style-type: none">• Intel Officer• Operations Officer• Logistics Officer
Sensor	The sensor provides the sensing function for IOS. The sensor can be any human or digital information provider.	<ul style="list-style-type: none">• Reconnaissance Team• Artillery Forward Observer• Counter-fire Radar
Friendly Unit	A friendly unit is an actor because it is external to IOS. IOS models friendly units, but friendly units take independent action and interact with IOS in many ways.	<ul style="list-style-type: none">• Infantry Company• Artillery Battalion• Mortar Platoon• Airplane

Table 7. IOS Actors

One or more IOS operators support each actor, and IOS servers are setup and maintained by IOS administrators. These roles are critical to correct operation of IOS, and their interaction with the IOS will be assumed for the remainder of this document.

Note that Friendly Units are actors, but in IOS, units are modeled as Tracks. Second, Enemy Units are not Actors, because in normal operation, there is no interaction between an Enemy Unit and IOS (they will not be entering data or initiating actions). Friendly Units can be sensors but because of its importance to the Command and Control process, the sensing function must be listed separately. Of course, sensors don't have to be units.

3. Essential Use Cases

Since there is no published IOS System Architecture, Use Cases must be inferred. Figure 11 below depicts three of the high-level essential Use Cases for the IOS. The IOS fulfills the requirements of these Use Cases. This evaluation will focus on the first two Use Cases.

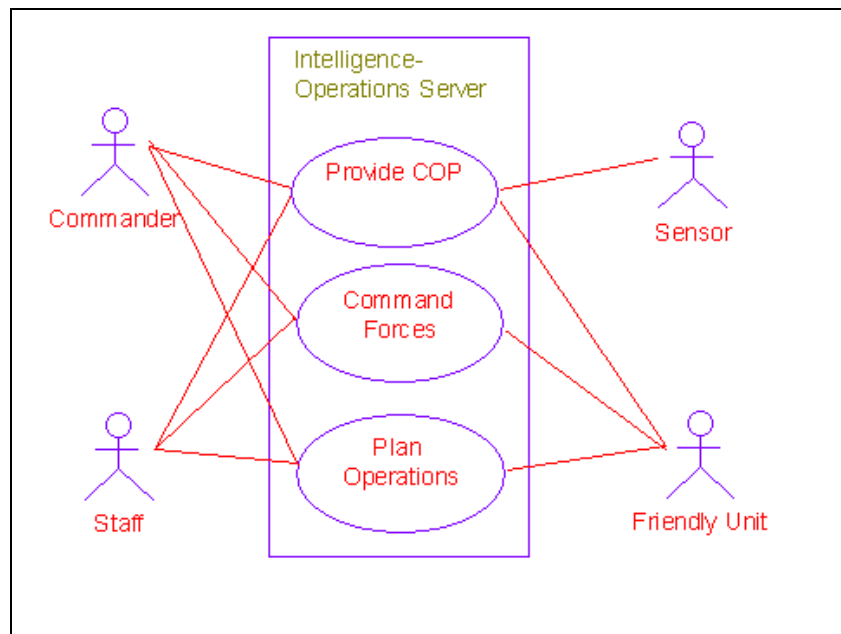


Figure 11. IOS Essential Use Cases

In the Use Case narratives below, the Requirement list is derived from Table 6.

USE CASE:	Provide COP
Actors:	Commander, Staff, Friendly Unit, Sensor
Purpose:	Use IOS to accurately model the Operational Area.
Overview:	This Use Case describes the steps necessary to prepare IOS for use. Prior to entering this Use Case, IOS station communications hardware must be set up. The user must understand which nodes he wants to connect, and have the detailed networking information needed in a typical TCP/IP network.
Type:	Primary and Essential
Cross References:	R1.1-1.6, R2.1-2.4, R3.1, 3.2, R4.1-4.4, R5.1-5.4, R6.1-6.5
Actor Action	System Response
1. This Use Case is initiated when a particular IOS server is powered on.	
	2. The system loads from an IOS CD-ROM, containing the DII COE Unified Build software and selected GCCS 3.x segments. Loading takes 45 minutes. The system presents a login screen.
3. The user logs in as sysadmin. The administrator sets up networking by using an "IOS configuration Wizard." Entries range from "Hostname" to "Primary DNS Nameserver."	
	4. The IOS takes the information and modifies the appropriate UB and GCCS segments and Solaris files to set up networking. Segment categories include communications, database, and track management segments.
5. The administrator configures the Track Database Manager (Tdbm), which is a server process residing on an IOS.	
	6. The Tdbm segment is updated with the master/slave configuration selected. The Tdbm segment begins processing data.

7. The user sets up more network and node data by editing the /etc/hosts and other critical files. The user can use Ping or other Solaris commands to verify connections. The User modifies the Defense Data Network (DDN) tables to set up network topology.	
	8. IOS is now ready to operate as a network server for COP data.
9. Client stations and sensors create COP objects (such as Tracks) and propagates them by transmission in any of a number of common message formats, including OTH-Gold. Every attribute of a Track can be modified as needed. Clients can filter COP data at the workstation using overlays.	
	10. Server segments process incoming messages and broadcast or unicast messages based on the network topology.
11. The senior Command becomes the Track Correlator (the “TOP COP”). The track correlator runs several segments that allow him to merge tracks and ensure data remains consistent within the network.	
	12. The IOS integrated databases (Tdbm) merge the tracks as required and presents the COP.
ALTERNATE Courses: Step 12: If the server becomes unavailable, a server monitoring process (the Joint Process Monitor) warns the client with an icon on the client’s screen.	

Table 8. Essential Use Case Provide COP (IOS)

USE CASE:	Command Forces
Actors:	Commander, Staff, Friendly Unit
Purpose:	Use IOS to send the information required to Command and Control Forces. This Command and Control information includes the tactical database, Operations Orders, plans, and overlays, messages, and alerts.
Overview:	This Use Case describes the steps involved in communicating Command and Control information from one node to another. The enemy situation is not part of this Use Case. The Use Case “Provide COP” must be complete prior to entry into this Use Case.
Type:	Primary and Essential

Cross References:	R1.1-1.7, R2.1-2.5, R3.1,3.2, R4.1-4.4, R6.1-6.3	
Actor Action	System Response	
1. This Use Case is initiated when a user on an IOS client machine (such as a workstation running C2PC software) creates an Operation Order, Plan, message, or alert. The user uses the Joint Mapping Toolkit (JMTK) to display mapping data. The user creates written products and map overlays as needed to produce an order.		
	2. The C2PC software provides an integrated set of tools to create the order.	
3. The user selects the recipients. If the client knows the address of the recipient, the address is added. Otherwise, the user enters the address, and sends the product.		
	4. The C2PC software logs the message and sends the message to the IOS. The IOS resolves the addresses and sends the products to the appropriate clients. Messages are sent in common format.	
5. The recipients receive the message on their client. If they have set the alert criteria, an audible alert is sounded signifying an incoming message.		
ALTERNATE Courses: None		

Table 9. Essential Use Case Command Forces (IOS)

4. COP Network

The IOS meets the requirements and Use Cases listed above using a system network overlaid on the military organization (see Appendix B for an example military organization). From the perspective of an IOS at the Regimental Combat Operations Center (COC), the IOS has relationships with other COC's. The relationships are defined by the information contained in the network routing tables. There is no knowledge of which nodes are "higher" or "adjacent," because the only information the IOS has about them is their name and network address. Actors overlay meaning on the nodes and set up broadcast and unicast forwarding patterns based on their perception of how information should travel. IOS does know, however, which is the Track Database Manager (Tdbm) Master station for COP synchronization, and that process is hidden

from the user in normal operation. There are other databases that have a similar relationship between IOS nodes, but for brevity they are not mentioned by name.

Given the scenario in Appendix B, there is only one IOS, and that is at the Regimental COC. The remainder are IOS clients. The regimental COC is the “TOP COP” responsible for track correlation and track management. This requires active staff participation via a “Track Correlator.” Figure 12 below depicts the IOS Network. In the Figure, the COP network is carried over UHF frequencies by the Enhanced Position-Location Radio System (EPLRS). This network is a point-to-point (unicast) network, but will be a multicast network in the near future. At each of the Battalions, one C2PC acts as a gateway for all the C2PC clients at the Battalion COC. The gateway is labeled as a C2PC GW. (At the Artillery Battalion, the COC is called the Fire Direction Center – FDC.)

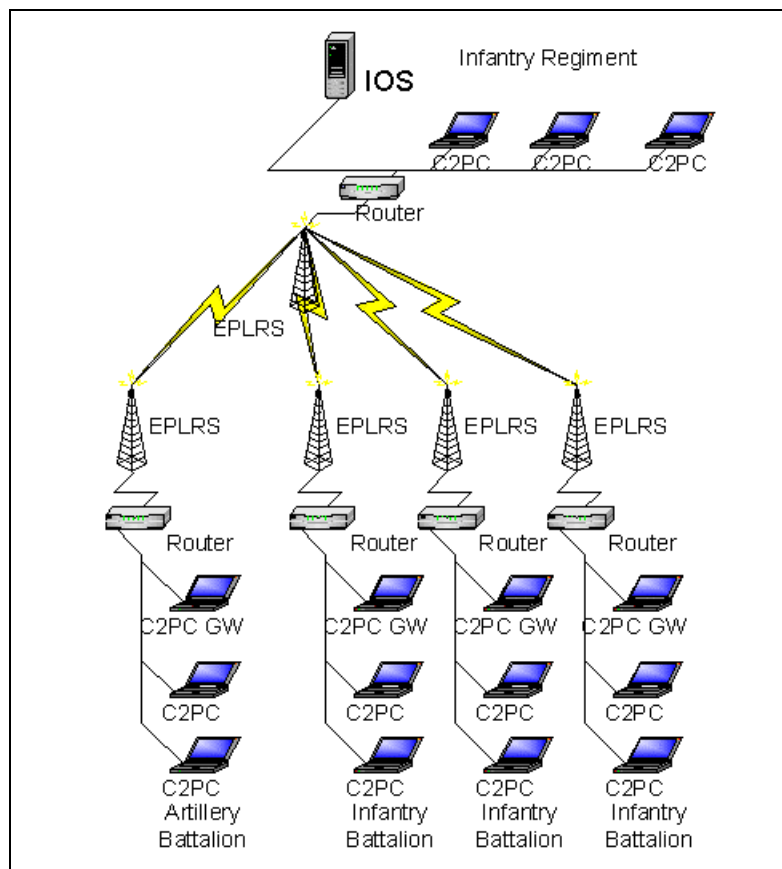


Figure 12. Regimental COP Network

5. Derived IOS Classes

The pertinent objects in the IOS model are graphically depicted in Figures 13 and 14 below. The first Figure shows the overall concept for management and display of the COP. There are more items in the COP that are not indicated and irrelevant to this discussion. In the Figure, a “View” corresponds to a CTP or CROP. An “overlay,” which has a collection of locations, modifies the COP to present the view. For instance, an overlay could be a unit boundary. The boundary would have an outline, with the outline being a collection of locations. The unit boundary could have properties set to show only unit tracks inside the boundary. In this example, no ELINT tracks, nor any other track besides unit tracks, would be displayed.

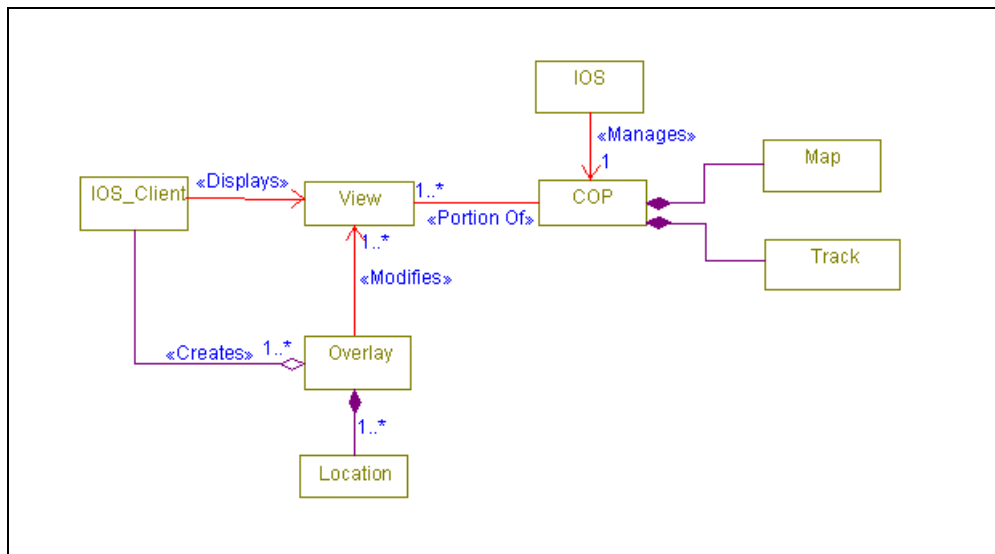


Figure 13. IOS COP

Figure 14 explores the concept of the Track. The “Track” has special significance in the GCCS and IOS world, as it is the primary mechanism of object management in the COP. Tracks are input into the COP either by manual entry at an IOS client or by detection by a sensor. A sensing is called a contact, which contains information about the Track and a position report at a given point in time. As shown in the figure, in IOS a track contains a collection of contacts. In this diagram, only the tracks relevant to ground operations are listed.

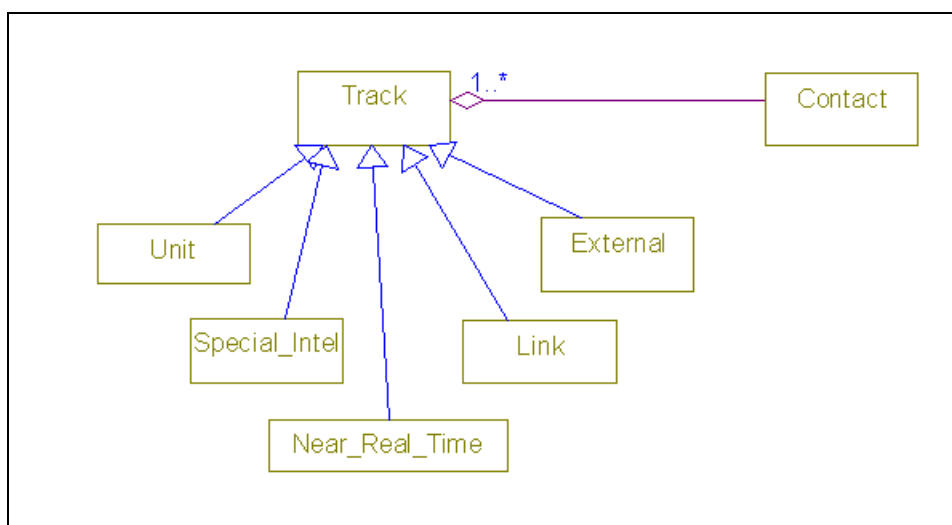


Figure 14. IOS Track Class.

The following Table lists all the major Track types defined in IOS, and their primary use. The first column, “Track identifier,” is a one-character code located at the first position in the Track Identification number (Track ID). The Track ID is used on the local machine to identify tracks. [CHBK95, 24]. The alert reader will notice that there are several inclusive Track identification schemes at work in the track classification scheme. Indeed, a track can have several different sources of contact information. For instance, a ground unit can have an ELINT or COMINT hit, but still be classified as a unit track. This can be a source of confusion. Further, the majority of tracks on any particular IOS will be external, but may not be so indicated in the Tdbm. Finally, several of these Track Types are for systems that are no longer deployed. The Track ID is an internal number. It does not necessarily bear any relation to the physical world object the track represents. In other words, there is no independent, objective way to identify Track ID’s from the characteristics of the item being represented. The one-character code may provide some information about the initial source of the report, and an operator can infer from the sensor what type of object it is. This ambiguity about the Track ID is a stumbling block to sharing track data with non-IOS systems such as AFATDS.

Track Identifier	Track Type	Purpose
A	Ambiguity	Track doesn't fit one of the other classes.
B	Acoustic	Produced from Acoustic sensors.
C	Special Intelligence/Comint	Produced from classified sensors.
E	Emitter/ELINT	Produced from classified sensors.
F	Submarine Fire Control Systems (FCS)	Produced from Submarine FCS sensors.
L	Link-11, Link-14, Link-16	Produced or received from Tactical Data links.
N	Near Real-Time	Produced from classified sensors.
R	RAYCAS(V)	Produced by a Raytheon shipboard Radar, first introduced in 1981.
S	SPA-25(G)	Produced by the SPA-25(G) radar sensor.
T	Platform	Represents a ship.
U	Unit	Represents a ground unit.
X	External	Received from an external source.

Table 10. IOS Track types. (After [CHBK98, 24-26]).

The preceding diagrams and discussion may have given the reader the impression that the COP is static. However, the COP is being updated all the time, from every sensor that feeds the system. Yet the processes that make that happen are in the background, other than for the “TOP COP,” who is charged with ensuring the COP is consistent given all current data. While it is not static, the COP as implemented on IOS is an information source rather than a decisionmaker, in that it provides information to the commander for him to act. In terms of the OODA loop, IOS doesn't “Decide” on the data, it only supports the commander in deciding and acting. This paradigm is different than that of AFATDS, which in “automated” modes will cause ammunition to fly downrange without human intervention.

6. Interoperability Requirements

Interoperability requirements were listed in the TCO ORD and COE in 1995. Specifically, these documents enumerate a requirement for TCO to be interoperable with all the DoD communication protocols and message formats then current. These communication standards include Over-The-Horizon – Gold (OTH-Gold) messaging, TCP/IP, and Ethernet for Local Area Network communications. The ORD also requires

interoperability with several named systems, to include AFATDS, IAS, JMCIS, the DACT, and GCCS. Again, “interoperability” is not defined. At least one Program Officer has interpreted “interoperability” to mean “Can exchange one or more bytes of information.”⁴ [KUBI01] This inadequate definition of critical interoperability requirements has led to inadequate implementation.

At this point, IOS is the de-facto “build-to” Marine Corps tactical system, known as the System of Record (despite receiving all funding from the TCO and IAS programs). IOS has reached its current stage via evolution. As a consequence, the IOS is the system to which other systems must interface, not the other way around. Since IOS has the Unified Build and a standard set of API’s, other system developers, such as those on the AFATDS project, have specified the AFATDS/IOS interface. Further, middleware vendors can write software that interfaces with a given version of IOS and can expect to have a reasonable amount of success – until either the IOS or other system version changes. Version changes happen about every eighteen months, while “patches” (minor software fixes) occur more frequently.

C. IOS IMPEMENTATION ISSUES

1. Software Development

IOS has no current Requirements Document, no requirement for artifacts such as help manuals or other documentation, and no development plan beyond what the current Project Officer proposes. The Operational Requirements were originally written for TCO software in 1995. With no current operational requirements document, it’s hard to hold system developer’s feet to the fire for not meeting requirements. SPAWAR Charleston has limited history on how the software got to where it is today, and the typical answer for this lack of information is that the system is really “GCCS-lite” – go talk to the joint people. Meanwhile, the Joint community is worried about the three different flavors of GCCS that they are required to interoperate with, and continually slipping back on their

⁴Captain Kubicki surmised this to be the requirement actually implemented, based on program decisions prior to his assignment as MARCORSYSCOM AFATDS project officer in 1999.

timelines for producing version GCCS 4.x (originally slated for release in 1999, and still not released.) However, based on reports from the Fleet Marine Force, IOS is a successful system in that it meets the requirements of the Use Cases presented above. The evolutionary strategy of combining software programs is a valid one for the Marine Corps. This permits the deployment of more capability using less hardware.

2. Support to the Fleet Marine Forces

A particularly vexing problem with IOS and similar systems is their complexity and lack of reliability. The requirement for extreme mobility produces cascading negative effects on the IOS. Many problems stem from mobile networking issues, but other issues are lack of robust hardware and the inability to connect reliably over long distances without wires. Second, the machines are not completely reliable. Although these systems have become more reliable, and identifying an exact source of troubles is difficult, the response of the Fleet has been to hire “TechReps” (contractors) to care for the machines in the field. While effective in the short term, this is not an optimum solution, because it exposes civilians to needless risks while forcing the Marine Corps to lose a boatspace for a combat-ready Marine. From a contracting point of view, the more systems are in the field, the more contractors from the original program are needed. In other words, four systems fielded results in Corps-wide costs for four contractors from four different companies.

3. Future Capabilities

IOS software is tied to GCCS. Therefore future capabilities will mirror that of the GCCS program. The next GCCS software version is version 4.x. GCCS 4.x has a quite different internal structure from Version 3.x. This different structure is one of the reasons the Initial Operational Capability date has slipped from 1999 to (possibly) early 2004. This version has the following planned capabilities:

- Ensure all GCCS 3.x functionality is retained.
- Ensure previous patches and fixes from version 3.x are supported.
- Provide a Microsoft Windows 2000 client.

- Provide a Microsoft Windows “look and feel.” This requirement decreases training costs.
- Support network and remote installation.
- Integrate an internal XML data scheme.

Tying IOS to GCCS is an appropriate choice for a system designed to manipulate the COP. IOS is DII COE level 8 compliant. What is more important is that the Marine Corps accurately define the requirements for the COP at the tactical and operational levels of war. The Corps must further ensure that these requirements are met not only in IOS, but GCCS as well. The issue of whether the GCCS data model (i.e., the “Track” paradigm) is sufficient for ground combat will be dealt with in the interoperability requirements chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

V. THE ADVANCED FIELD ARTILLERY TACTICAL DATA SYSTEM (AFATDS)

AFATDS is the digital C2 Program of Record for the fires functional area. It coordinates employment of ground, air, and sea based fires to support maneuver units. AFATDS analyzes available fire support assets and applies commander's guidance to attack targets based on an optimal fire support solution. AFATDS was designed for and fielded to the tactical and operational levels of the U.S. Marine Corps. Therefore, AFATDS is fielded from the artillery battery to the artillery regiment and at the supported infantry headquarters from battalion to Marine Expeditionary Force (MEF).

AFATDS is the latest step in fire support C2. Among the earliest uses of computers was the calculation of firing tables for artillery projectiles. Nothing in these older systems could be considered "user-friendly," and the artillery community learned to adapt the man to the machine in order to get the desired result of accurate predicted fires. The history of the development of fire support systems is germane to the current AFATDS system, as it provides context for the requirements for the AFATDS system.

A. HISTORY

1. The Gunnery Problem

In order to ensure accurate predicted indirect fires, five elements must be accounted for in the solution. These elements are: accurate information about the projectile and propellant, accurate weapons information, accurate target and weapon locations, accurate meteorological information, and accurate computational procedures. From the earliest days of artillery, mastery of these five elements ensures effective fires on the enemy. Early automated solutions focused on "accurate computational procedures," but as systems continue to grow in power and shrink in size, automated systems increasingly are involved in the other four elements, which require accurate sensors. These five elements together are termed "technical fire direction," as they contribute to actually getting steel on target.

A further goal of automation is the solution of “tactical fire direction.” Tactical fire direction solves the problem of fires employment and answers the questions of why are we firing, who will fire, what type of fires needed, and how much fires are needed for a given level of effect. Tactical fire direction can be summed up as “Decide, Detect, Deliver, Assess (D3A).” First, the commander decides what are his priority targets. Then surveillance assets (sensors, including observers) detect those targets. Once detected, the appropriate type and amount of fires for the desired effect is delivered to the target. Finally, the effects are assessed and the cycle is repeated if the desired effects have not been achieved. Tactical fire direction requires much more information from many more widely dispersed sources than that needed for technical fire direction, and is the harder problem to solve.

2. Field Artillery Digital Automatic Computer (FADAC)

FADAC was developed in 1959 by Autonetics, a subsidiary of North American Aviation, Inc. FADAC was the first deployable digital system designed to accurately solve the technical fire direction problem (See Figure 15). An operator in the Fire Direction Center (FDC) would enter data by using a matrix of rows and columns of switches, with values stored at the intersections. Meteorological data could be entered using punched paper tape. Once all the data was stored (consisting of the other four elements of accurate predicted fires), pushing a button solved the differential equations for the projectile, and weapon-aiming information was displayed “in decimal form.” [BRLA61, 254] Apparently, the designers thought the display of decimal numbers over binary or hexadecimal numbers was a selling point.



Figure 15. FADAC terminal. (From [BRLA61, 254])

FADAC relieved the artillery community of several laborious and error-prone steps in calculating firing data, and greatly improved field artillery support in Vietnam. [DAST92] Yet, FADAC left a lot to be desired. The operator had no way to check for incorrect data entry, so manual methods were used as a backup to verify the automated solution. There was no automated communication to the firing platform, and limited communication between various command and control nodes. FADAC had no method for conducting Tactical Fire Direction. FADAC was common in Army and Marine units from 1960-1980.

3. TACFIRE

TACFIRE (an acronym derived from TACTical FIRE direction) was developed in the late 1960's by Litton Data systems as technology continued to progress. [LITT00] Besides using the newest digital technologies to reduce weight, power consumption, etc., TACFIRE was the first system to network the various Fire Support agencies such as the FDC and Fire Support Coordination Center (FSCC). TACFIRE automated several of the manual processes for both technical and tactical fire direction. As fires generally have to be cleared by the unit responsible for the area in which the fires are called, this automation greatly speeded the delivery of fires. TACFIRE digitized information from radars and meteorological stations, again providing a direct benefit to the field artillery. [DAST92] TACFIRE was deployed at various levels in the Army from 1977 – 1996.

However, problems with TACFIRE kept the Marines from acquiring it. TACFIRE was a large system, straining transportation networks. It was not designed to operate while moving, a critical requirement for the Marines. TACFIRE used a proprietary, fixed format, character-oriented message set.⁵ Communications rates were slow - between 150-2400 bps. Then current radios were designed to support voice communications. Because of the number of the TACFIRE devices deployed, other devices were required to conform to the TACFIRE communications standard for TACFIRE communications. The Forward Observer (FO) still did not have a method of entering missions digitally, requiring him to use voice circuits to the battery FDC to send in fire missions. Finally, although the TACFIRE interface was better than FADAC, few human factors were considered.

4. BCS/LTACFIRE/IFSAS

Because of its size, the Marines did not field TACFIRE. Instead, the Marines acquired the Battery Computer System (BCS) and began a parallel development effort called the Marine Integrated Fire and Air Support System (MIFASS). After a short while, the technical complexity of the Fire Support challenge became clear, and the Marine Corps settled for the capabilities of the BCS. The BCS was the portion of TACFIRE residing at the firing batteries that solved the problem of technical fire direction. The BCS calculated firing data for each gun in the battery, producing better target effects. The BCS communicated digitally with other BCS computers, TACFIRE devices, Forward Observer devices, and the gunline. The BCS could continue to operate while moving. The prime contractor was Litton Data Systems, while Telos Corporation wrote the software in Ada. The Corps also procured several handheld computers for use as a backup to the BCS.

Meanwhile, in the mid eighties the Army began deploying Light TACFIRE, which was TACFIRE software ported to smaller hardware, for their light divisions. Operations in Southwest Asia were conducted with a mix of TACFIRE and LTACFIRE, interoperating with TACFIRE protocols over voice radios. After Desert Storm and as

⁵ Later versions supported an improved Variable-Format Message set. [WATK02]

hardware continued to get cheaper and more powerful, the Army and Marine Corps began fielding the Interim Fire Support Automated System (IFSAS). IFSAS was designed to be a stopgap C2 system until AFATDS could be fielded. IFSAS was again TACFIRE software ported to a new hardware platform known as the Lightweight Computer Unit (LCU).

5. AFATDS

During the late 70's, the Department of the Army began looking for a replacement for TACFIRE. Specific goals were to effectively apportion limited fire support assets to targets, integrating the scheme of maneuver with fires to produce the largest effects on the enemy. In effect, this would completely automate the tactical fire direction process. In 1981, the Army approved a plan to develop the Advanced Field Artillery Tactical Data System, with fielding scheduled for the early 1990's. [DAST92] It is interesting to note that the Army, thinking ahead, planned so much time for AFATDS development. Yet in the end, even this amount of time was insufficient for fielding. The first version of AFATDS software was released in 1996.

The initial AFATDS contract was awarded to Magnavox Electronic Systems, Fort Wayne, Indiana. Development of AFATDS was anything but smooth. The Field Artillery community wanted a system that would meet all the requirements of a high threat, target rich scenario such as the defense of NATO. Such a defense would require automated tactical fire direction of a scale unseen in then current fire support systems. [WATK02] Therefore, there were extensive requirements to automate the entire fire support problem. Further, technology was moving along at a rapid pace, and as the Field Artillery community saw better emerging technologies, they wanted more capabilities.

There were also significant design challenges. First among the challenges was the marked lack of bandwidth in the tactical arena. The majority of communications at the Marine Regiment and below is via FM voice radio, which is not optimal for digital communications. Second, each AFATDS node is required to maintain common databases in order to ensure synchronization between nodes, requiring significant computational power and complicating replication issues. Finally, automated fire

mission processing required digitized sensors and shooters in a networked environment, which is a significant task even with perfect communications. One 1994 report from the Department of Defense Inspector General's Office was particularly damning:

The AFATDS program is not ready to proceed into the production and deployment phase of the acquisition process. The AFATDS software to be deployed lacks critical capabilities necessary to fulfill user requirements, including communication with other user systems. Subsequent versions of AFATDS software, potentially capable of meeting user requirements, do not have a dedicated engineering and manufacturing and development phase to achieve production hardware and software configurations suitable for deployment. As a result, the Army could spend \$187.2 million for hardware that does not meet requirements, spend \$4.6 million for an initial operational test and evaluation that will not prove AFATDS ready for fielding, experience further delays in the development of software, field software that does not meet user requirements, and support two systems [IFSAS and AFATDS] to accomplish the same mission. [DDIG94]

Despite this negative assessment, the Artillery community continued to support AFATDS, as there was no other choice. To help mitigate the problems mentioned above, the Program Manager and system developers agreed on an iterative software development model, instead of the waterfall model used previously. Each iteration would offer more functionality than the last. The program was also helped by advancing technology, both in communications systems and in the computing power available.

Although the software versions were originally numbered starting with version 1, they were changed to reflect the year of expected release. Version A96 was the first fielded, followed by versions A97 and A98. These versions were also delayed such that there is now no longer a correspondence between the version number and the year of issue. Version A98 (the version shown on the display in Figure 16 below) is the fielded version. Version A99, which is being fielded now, will be the first version to offer technical fire direction, replacing the BCS at the Firing batteries. Despite the painful development history of AFATDS, the Field Artillery community has come to view AFATDS as an essential fire support system.

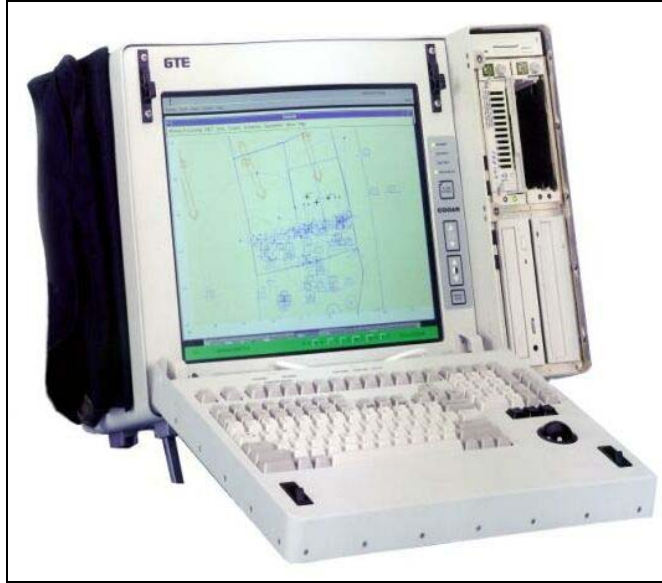


Figure 16. AFATDS Workstation (From [APIC02])

B. OPERATIONAL REQUIREMENTS

For various reasons, both operational requirements and software versions change frequently. The process used to list the requirements will be to list a subset of the user requirements derived from the version 2.1 (A99) System Segment Specification [FSSS00], and the U.S. Marine Operational Requirements Document [AORD00], list the system actors, derive the Essential Use Cases, and provide collaboration and class diagrams. Operational requirements are better understood in the context of the Marine Corps tactical organization for combat. For readers unfamiliar with U.S. Marine Corps Organization, Appendix B clarifies U.S. Marine Fire Support relationships and lists one scenario where AFATDS and IOS could be used.

1. Selected User Requirements

AFATDS is a complex system – the User’s manual is in four volumes with nearly 1000 pages. [AHL99] Therefore, the following table provides a small but relevant subset of the AFATDS system requirements as articulated in Chapter 3 and Appendix I of the AFATDS Version 2.1 System Segment Specification. [FSSS00] These requirements were selected based on relevance to the Fire Support mission. The “requirement number”

listed here does not correspond to any AFATDS artifact, and is used for tracking the requirement in this document.

User Requirement	AFATDS Process	Requirement Number
Communicate Digitally	<ul style="list-style-type: none"> Establish Communications Manage Alerts Autoforward Messages Filter Incoming/Outgoing Messages Unicast/Broadcast Data 	R1.1 R1.2 R1.3 R1.4 R1.5
Maintain Accurate Friendly Unit Information	Create/Edit <ul style="list-style-type: none"> Friendly Unit Information Friendly Geometry Information Friendly Unit Disposition 	R2.1 R2.2 R2.3
Maintain Accurate Enemy Unit Information	Create/Edit <ul style="list-style-type: none"> Enemy Unit Information Enemy Unit Geometry Information Enemy Unit Disposition 	R3.1 R3.2 R3.3
Maintain Accurate Battlespace Information	Create/Edit <ul style="list-style-type: none"> Battlefield Geometries FSCM Geometries 	R4.1 R4.2
Develop and Apply Commander's Guidance (DECIDE)	Create/Edit Guidance Data <ul style="list-style-type: none"> Attack guidance Target Guidance Command and Control Guidance Trigger Event Criteria 	R5.1 R5.2 R5.3 R5.4
Process Combat Information (DETECT, ASSES)	<ul style="list-style-type: none"> Generate Targets Process Targets Filter Targets 	R6.1 R6.2 R6.3
Deliver Fires (DELIVER)	Attack Analysis <ul style="list-style-type: none"> Maintain Unit List Determine Mission Requirements Perform Geometry Checks Determine Recommended Attack Option Perform Mission Coordination Conduct Mission <ul style="list-style-type: none"> Determine Controlling Unit Compute Ballistic Data 	R7.1 R7.2 R7.3 R7.4 R7.5 R7.6 R7.7

Collect Combat Information (DETECT, ASSES)	Interoperate with External Systems Data Distribution Create/Edit Enemy Unit Information	R8.1 R8.2 R3.1-3.3
---	---	--------------------------

Table 11. AFATDS User Requirements (After [FSSS00])

2. Actors

In the UML, actors are the people or systems that interact with Use Cases. In the military, and thus in AFATDS, the same sequence of Actors is repeated for each level of the military hierarchy. For example, there is a Battalion Commander and a Regimental Commander. These are specializations of the Actor Roles. AFATDS makes distinctions between the various specializations when it is pertinent to completing the actions in the Use Cases. Specifically, distinctions are made in the Fires Coordinator, Observer, and Firing Agency Roles. The following table lists the characteristics of the principal Actors in the AFATDS system.

Actor Role Name	Description	Example Instances
Commander	The commander role is that of the person given responsibility and authority to prosecute the campaign. The commander sets fire support policies that are then encoded into AFATDS. The commander expects the information and data in AFATDS to be correct, and looks to AFATDS for situational awareness of fires issues.	<ul style="list-style-type: none"> • Battalion CO • Regimental CO • Division CO
Fires Coordinator	A member of the commander's staff fills the Fires Coordinator role. The Fires Coordinator is responsible for implementing the Commander's policy. He ensures that targets are correctly identified and assigns fires based on mission priorities. He provides human oversight of AFATDS operations.	<ul style="list-style-type: none"> • Artillery Battalion Liaison Officer • Artillery Battalion Operations Officer • Fire Direction Officer
Observer	The observer provides the sensing function for AFATDS. The observer interfaces directly with AFATDS using a message entry device and digital communications	<ul style="list-style-type: none"> • Artillery Forward Observer • Counter-fire Radar

	equipment. If digital equipment is not available, the observer can call for fire using voice transmission over radio, and an AFATDS terminal operator will enter the mission.	
Firing Agency	Any asset capable of delivering ordnance.	<ul style="list-style-type: none"> • Artillery Battalion • Mortar Platoon • Naval Gunfire Ship • Airplane

Table 12. AFATDS Actors

Each actor is supported by one or more AFATDS operators, and AFATDS workstations are setup and maintained by AFATDS administrators. These roles are critical to correct operation of AFATDS, and their interaction with the AFATDS will be assumed for the remainder of this document. It is important to note that a Firing Agency is considered an actor because it is outside AFATDS. However, inside AFATDS, this actor is modeled as a “friendly unit” object with high fidelity.

3. Essential Use Cases

Figure 17 below depicts four high-level essential Use Cases for the AFATDS system. AFATDS currently fulfills the requirements of these Use Cases. The first three Use Cases are modeled.

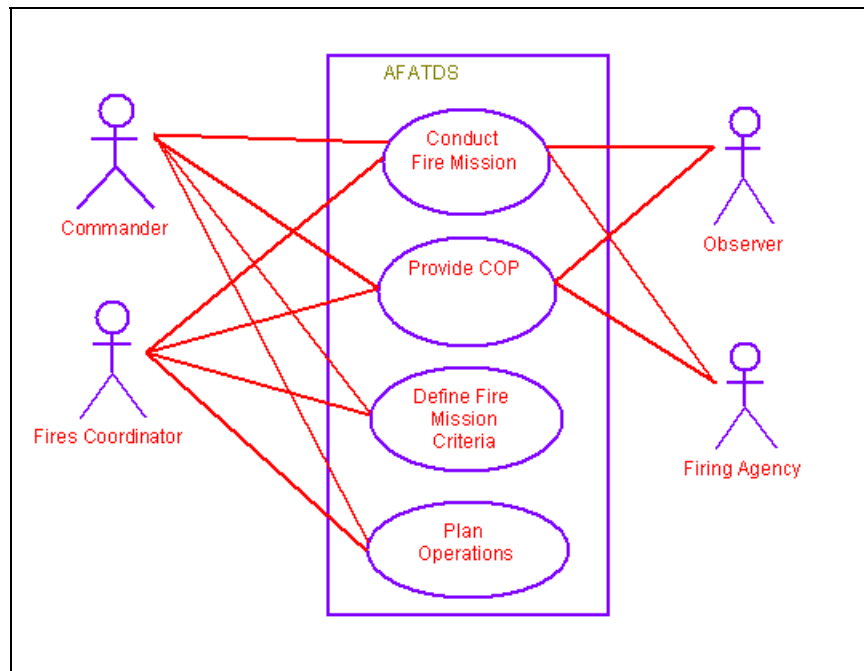


Figure 17. AFATDS System Level Use Cases.

The three most critical Use Cases are listed in the tables below. These Use Cases fulfill the Requirements set forth in the Requirements noted above (Table 11), as noted in the Use Cases.

USE CASE:	Provide COP
Actors:	Commander, Fires Coordinator, Firing Unit, Observer
Purpose:	Use AFATDS to accurately model the Operational Area.
Overview:	This Use Case describes the steps necessary to prepare AFATDS for use by establishing the tactical database. Prior to entering this Use Case, AFATDS station communications hardware must be set up.
Type:	Primary and Essential
Cross References:	R1.1-1.4, R2.1-2.3, R3.1-3.3, R4.1-4.3
Actor Action	System Response
1. This Use Case is initiated when a particular AFATDS workstation is turned on.	
	2. The first AFATDS powered on in an Operating Facility (OPFAC) becomes the master AFATDS. AFATDS loads the previously stored OPFAC configuration and databases.

3. The operator verifies the previously stored OPFAC configuration or changes the data if needed. The operator chooses the role of this particular OPFAC from the following menu: FSE/FSCC; FA CP/FDC; FU (Firing Unit); IUC (Independent User Center).	
	4. AFATDS takes the new configuration data. AFATDS loads the appropriate software to meet the requirements of the chosen role. If the role selected is IUC, AFATDS loads only a portion of program software.
5. The operator edits the Master Unit List (MUL), if desired. The MUL is the listing of all friendly units in the theater that have the capability to communicate with AFATDS, whether they are AFATDS machines or machines designed to interface with AFATDS. The MUL contains unit names along with their identifying information and communications parameters.	
	6. AFATDS changes the MUL.
7. The operator enters the authorized system users.	
	8. At this point, the AFATDS master station allows other AFATDS workstations to power up. AFATDS allows other users to login.
9. The operator assigns one or more duties to that particular AFATDS machine from the following menu: System Administrator, Communications Administrator, Message Monitor, Mission monitor. The operator creates communications and distribution lists that meet the requirements of the tactical scenario. The operator enters friendly unit information of the units controlled by that OPFAC.	
	10. Each AFATDS assumes the duties required. The AFATDS OPFAC gleans information about higher, adjacent, supporting, and supported unit relationships based on the operator entries.

11. The AFATDS operator enters geographic information and battlefield and enemy data as needed to update the common operational picture.	
	12. AFATDS takes this data and correlates it with other incoming COP data from other Operating Facilities. Friendly data is correlated using the MUL. AFATDS data is shadowed as necessary between master and slave workstations in the same OPFAC. AFATDS displays a view of the COP when requested. The user can change the view by filtering the data.
<p>ALTERNATE Courses:</p> <p>Step 1: If an AFATDS machine is not the first powered on in an OPFAC, it becomes a slave machine and AFATDS automatically shadows databases. Slave machines cannot edit the MUL or perform other system administration functions. Slave machines can execute steps 1 and 8-12 of this Use Case.</p> <p>Step 3: If the AFATDS operator changes the unit information, AFATDS loads a default database. This can cause significant disruption if the operator expected all other tactical data to remain the same.</p> <p>Step 5. The MUL must be the same across all AFATDS at all OPFACs. The AFATDS developers recommend that only one unit (the senior unit) modify the MUL and that all other AFATDS units import that MUL from a disk.</p>	

Table 13. Essential Use Case - Provide COP (AFATDS).

USE CASE:	Define Fire Mission Criteria
Actors:	Commander, Fires Coordinator
Purpose:	Prepare AFATDS to automatically process fire missions
Overview:	This Use Case describes the steps necessary to prepare AFATDS for mission processing by establishing Commander's Guidance. Prior to entering this Use Case, an AFATDS workstation must be initialized as described in the Use Case "Provide COP." In particular, the MUL must match among all AFATDS stations. Further, the AFATDS role must be defined.
Type:	Primary and Essential
Cross References:	R4.1-4.4, R7.1-7.2
Actor Action	System Response
1. This Use Case is initiated when the Commander defines Fire Mission Criteria. These criteria are stated in the form of fire mission processing rules and Commander's guidance for fires. Typical guidance includes fire mission type priorities and targeting priorities by target class (i.e., fire on threat C2 nodes immediately when identified but don't shoot supply sites). The operator enters this information into AFATDS.	
	2. AFATDS takes the data as guidances and rules, and stores it into multiple "mission selection criteria" data tables.
3. The operator enters types and locations of friendly fire support units. The Fire Support Coordinator enters amounts and types of ammunition available by unit.	
	4. AFATDS takes the data and stores it into multiple "firing unit" tables.
5. The operator defines battlespace geometry in terms of restrictive and permissive FSCM's, by location.	
	6. AFATDS takes the data. During the Use Case "Conduct Fire Mission", it will check the target grid against this data for possible violations of geometry constraints.
7. The operator enters all current observer locations. The operator defines how observers will connect to the AFATDS network.	

	8. AFATDS takes the data and stores it for use. Observers are treated as units by AFATDS.
<p>ALTERNATE Courses:</p> <p>Step 4: If other AFATDS OPFACs have unit information, after a communications path has been established, the OPFACs will update unit information between them. This includes information about location and ammunition available.</p>	

Table 14. Essential Use Case – Define Fire Mission Criteria. (AFATDS)

USE CASE:	Conduct Fire Mission
Actors:	Observer, Fires Coordinator, Firing Unit
Purpose:	Deliver fires on targets.
Overview:	This Use Case describes the steps necessary for AFATDS to deliver fires on targets of opportunity. This Use Case describes the primary functionality of AFATDS.
Type:	Primary and Essential
Cross References:	R1.1-1,3, R2.1-2.3 R5.1-5.3, R6.1-6.7, R7.1-7.2
Actor Action	System Response
1. This Use Case is initiated when an Observer calls for fire using a digital device. The observer has a wide range of choices. He chooses to call a “Fire For Effect-When Ready” mission. This means that the firing battery is responsible for firing when it is ready, not at any command of the observer.	
	2. The receiving AFATDS OPFAC verifies validity of the Call for Fire Grid and other data. AFATDS assigns a target number to the target. AFATDS compares the “suspect target” to target selection standards and determines attack precedence. AFATDS filters the target, checking for target duplication. AFATDS routes the Call for Fire to a Fires Coordinator OPFAC based on target location and other properties.
3. The Fires Coordinator checks the target location against known friendly locations and other parameters. The Fires Coordinator AFATDS system approves or denies the Call for Fire.	

	4. AFATDS determines which unit can range the target, has the appropriate ammunition, and is available for missions. AFATDS sends the mission to that Firing Unit. AFATDS sends a message to observer telling the Observer what to expect from which Firing Unit.
5. The Observer acknowledges receipt of the message to observer.	
6. The Firing Unit acknowledges receipt of the mission.	
	7. The Firing Unit AFATDS process the Fire Mission by conducting technical Fire Direction and sending the appropriate commands to the gunline. The guns fire.
8. The observer observes the effects on the target and ends the mission. The observer sends a message containing tactical intelligence about the effects on the target.	
	9. AFATDS sends an End of Mission message to the Fires Coordinator and the Firing Unit. AFATDS saves the tactical intelligence from the mission and updates the COP. AFATDS updates the Firing Unit information by decrementing the amount of ordnance expended from the Firing Unit.
<p>ALTERNATE Courses:</p> <p>Step 1: If the observer doesn't have a digital device or loses digital connectivity, the observer will use voice transmission to reach the appropriate AFATDS operator, who will enter the mission into AFATDS. In either case, the AFATDS processing steps are the same.</p> <p>Step 3: This step can be done automatically or manually. Current USMC practice is to have a "man in the loop," so this check is shown as an Actor action, with communication via the AFATDS system.</p> <p>Step 4: If the Mission is Denied, AFATDS sends a Message to observer stating that, and saves the target for later processing or analysis.</p> <p>Step 8: The Observer can choose to adjust the fall of shot based on demonstrated lack of effect on the target, or can repeat the mission if more effects are needed. In these cases, processing loops to step 4.</p>	

Table 15. Essential Use Case – Conduct Fire Mission (AFATDS)

4. AFATDS Network

AFATDS meets the requirements listed above using a system network overlaid on the military Fire Support Structure (see Appendix B for an example scenario). From the perspective of any OPFAC, the OPFAC has relationships with higher, adjacent, supporting, and supported nodes. These roles are not required to be filled (the highest node has no “higher” relationship), but when filled, define what messages are sent and when they are sent. Although AFATDS is truly a “Fire Support” system, it has traditionally been viewed as an artillery-specific system, and therefore artillery serves as the example agent providing fires. Typical OPFACS are the infantry Regimental COC and the Artillery Battalion FDC. Figure 18 below shows the AFATDS logical network that would be used for the scenario listed in Appendix B. Using this example, there are 9 FO teams, 3 Infantry Battalion FSCC’s, one artillery Battalion FDC, and three battery FDC’s.

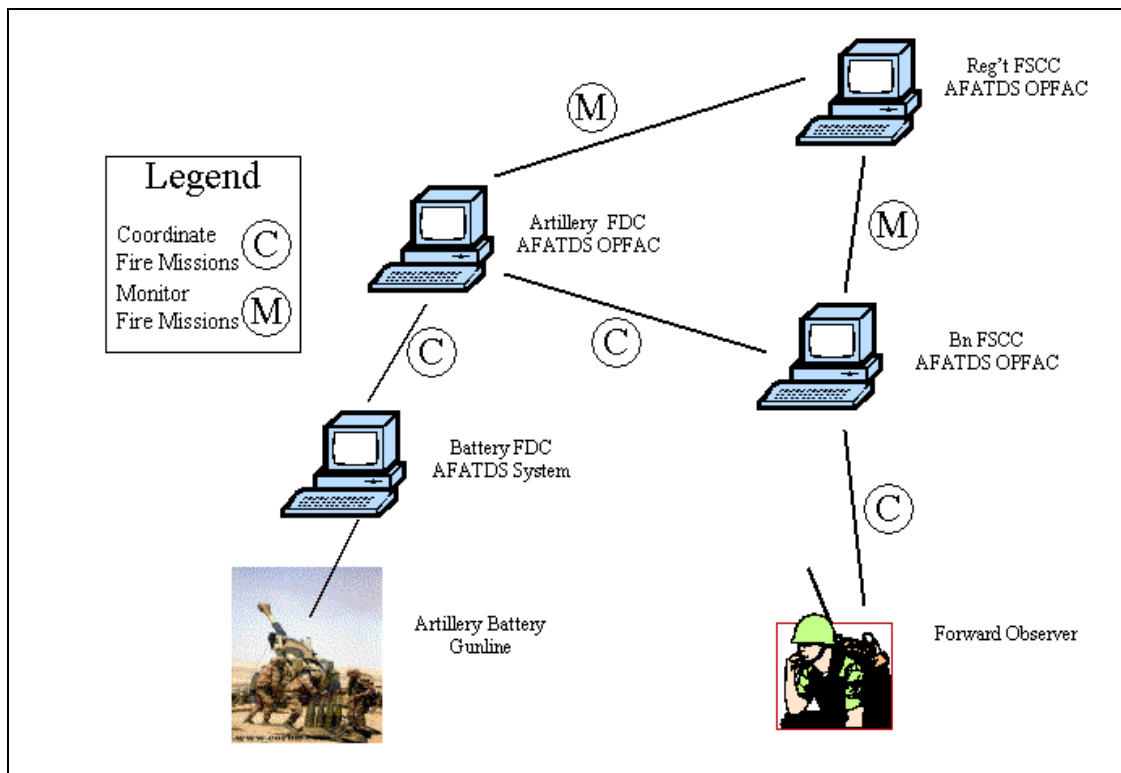


Figure 18. Example Infantry Regimental AFATDS logical Network.

5. Derived AFATDS Classes

Recall from chapter III, a Class is the term for objects that share the same characteristics. For example, FriendlyUnit is a class, while a particular friendly unit (say 3rd Battalion 11th Marines) is an object (also called an Instantiation) in the class. Classes can have attributes and methods, where attributes describe objects and methods are the activities that the class participates in. In AFATDS software, class attributes are the row and column names stored in a relational database, while the values of a given attribute are stored at the intersection of the rows and columns in the database. Methods are not explicitly given to classes, since AFATDS was originally not designed using object orientation. The purpose of using classes to describe AFATDS is that it allows a more explicit picture of the order of operations to be derived in the sequence diagrams that follow. The following diagrams graphically depict the classes derived from the Use Cases above.

Diagrams 19 and 20 below depict Classes involved in the Use Case “Provide COP.” Of note on these diagrams is the change in an AFATDS workstation to an AFATDS OPFAC while powering up. This change limits functionality while preparing the AFATDS to perform the appropriate OPFAC role in mission processing. Next, in AFATDS, an Enemy Unit is a separate concept from a target. In fact, a target can be any location or set of locations, no matter what is actually there. A Suspect Target is unprocessed, incompletely specified, or found not to meet target selection standards. Also note that in AFATDS, a geometry is 30 or fewer locations.

Enemy units are processed and viewed differently than friendly units, and Firing units contain significantly more information than other units in order for AFATDS to pick an appropriate firing unit, manage Firing Unit ammunition, and control movement.

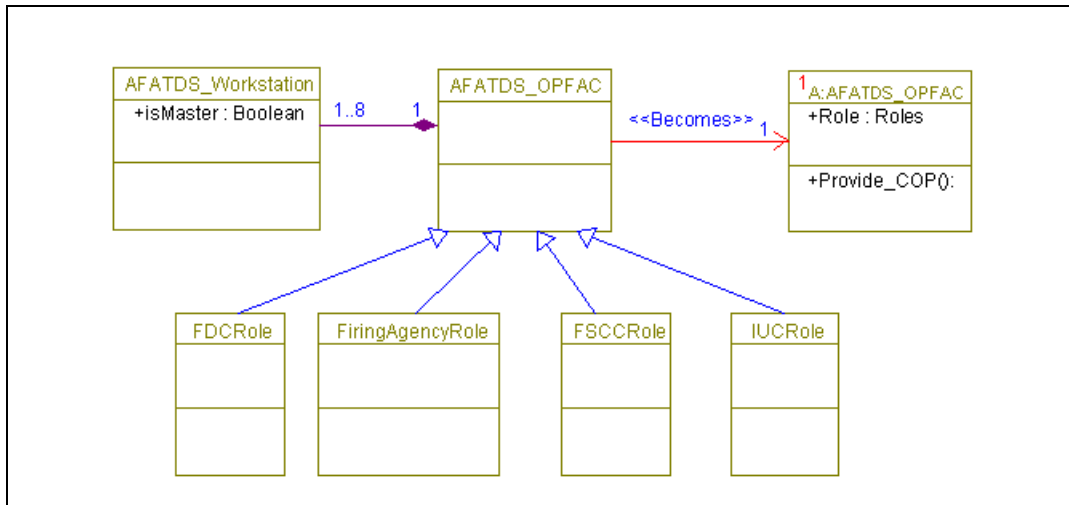


Figure 19. AFATDS OPFAC Class

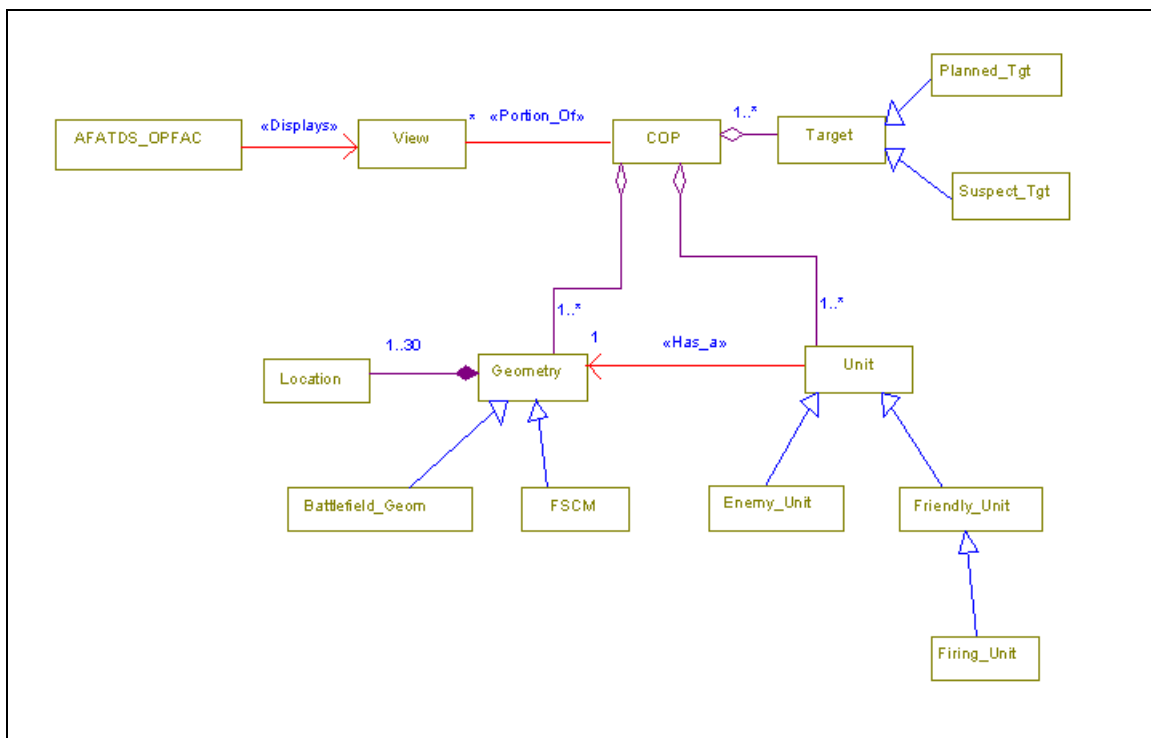


Figure 20. AFATDS Provide COP

The new concepts of guidance introduced in the Use Case “Define Fire Mission Criteria” are shown in Figure 21. In AFATDS there are actually 6 different types of guidance, but these three are the ones most used for mission processing. Important concepts here are that AFATDS processes fires for all types of Fire Support Assets, and

that attack guidance is contained in the FS_Atk_Guidance (“FS” stands for Fire Support). FA_Atk_Guidance represents Field Artillery specific attack guidance, while Target_Guidance provides the criteria for target processing. In the D3A cycle, target selection and processing is the key element in ensuring the Commander’s Intent for fires is met.

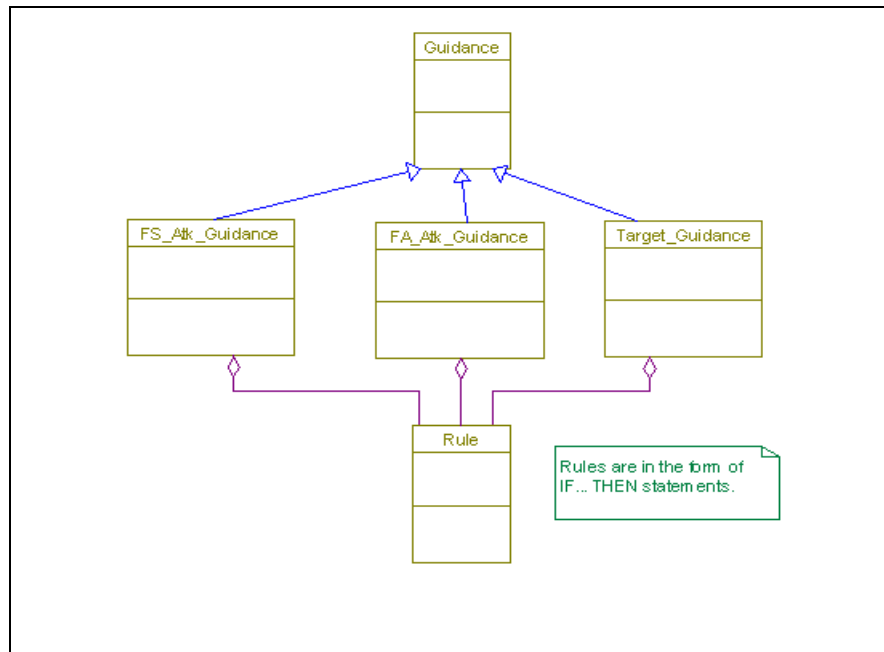


Figure 21. AFATDS Guidance Classes

The final Use Case, “Conduct Fire Mission,” does not introduce any new classes. It will be treated in the next section using a UML Sequence Diagram.

6. Use Case “Conduct Fire Mission” Sequence Diagram

As discussed in the UML Chapter, a UML sequence diagram emphasizes the actions a system takes to accomplish a task. The sequence of interactions between objects establishes a clear timeline. Although events may occur concurrently, event start and end times are explicit. The Use Case “Conduct Fire Mission” is a prime candidate for this treatment. No new classes are introduced. In a sequence diagram, Actors are represented outside the system boundary, while objects (not classes) are inside the system boundary. In this Use Case, the “system” is the complete AFATDS network. The pertinent actors are the observer and the Firing Agency. To simplify the Use Case, we

will assume that no operator intervention criteria have been set, i.e., AFATDS is in automatic processing mode. The Fires Coordinator and Commander both have contributed to the COP and guidance, but their actions are performed by AFATDS when AFATDS processes missions automatically.

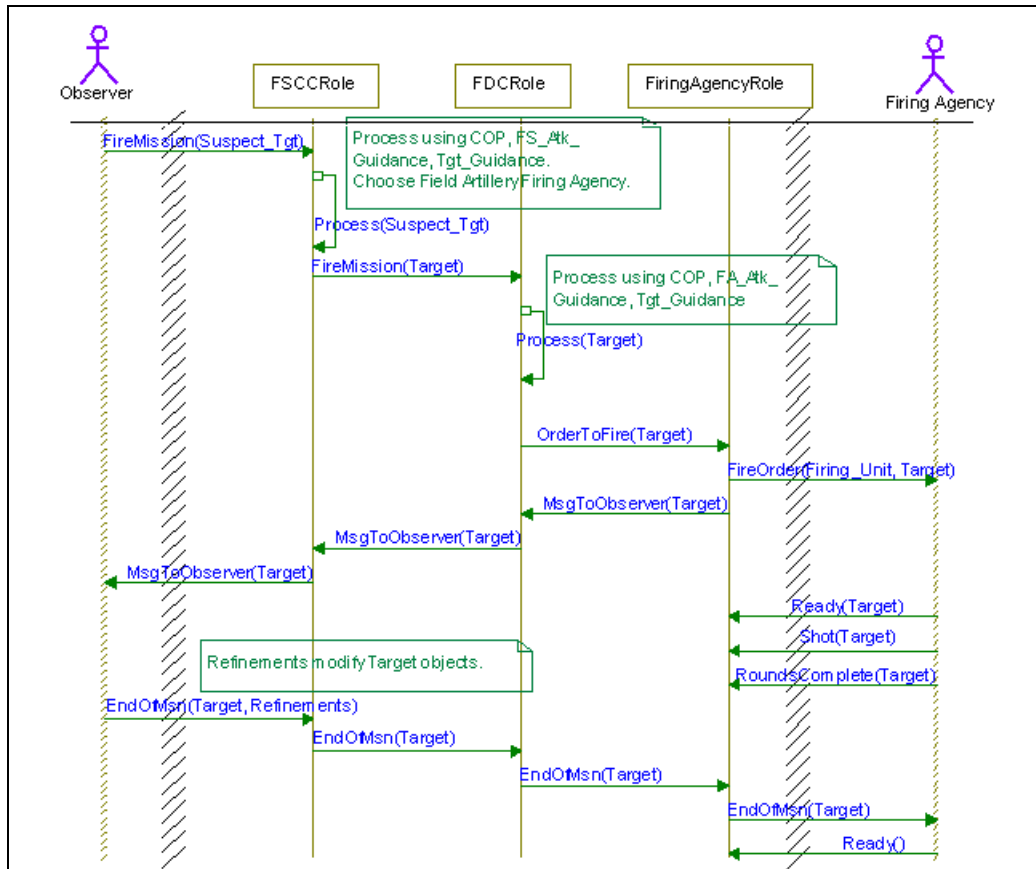


Figure 22. Use Case “Conduct Fire Mission” Sequence Diagram

The observer starts the action by sending a Fire Mission on a Suspect Target. The FSCCRole AFATDS processes the mission using the guidances indicated in the note in the Figure. Processing is fast – usually a few milliseconds up to a second. When the Suspect Target is approved, it becomes a Target, and the FSCCRole AFATDS sends a fire mission to the FDCRole AFATDS with the target data. The FDCRole AFATDS sends an “Order to Fire” to the FiringAgencyRole AFATDS at the Firing Unit (note the firing unit is outside the system boundary to the right). The FiringAgencyRole AFATDS sends fire commands down to the gunline. The FiringAgencyRole AFATDS sends a message back to the observer telling him what to expect (number of rounds,

target number, and other information). This message is repeated through the network back to the observer. The “Ready,” “Shot,” and “Rounds Complete” messages are internal to the firing agency processing with the FiringAgencyRole AFATDS, but the guns are firing. The observer sees the fall of the rounds, and decides to end the mission. The “End of Mission” message he sends is echoed throughout the AFATDS network, and mission processing is complete. The Firing Unit sends a “Ready” message back to the AFATDS network, signifying readiness for further missions.

C. INTEROPERABILITY REQUIREMENTS

1. Overview

When AFATDS was being developed in the 1980’s, the biggest interoperability concern was with legacy systems such as IFSAS and TACFIRE. The majority of ORD interoperability requirements addresses these fire support-specific systems. In the 1980’s, AFATDS was the cutting-edge in the ground C2 arena, and other C2 systems were either undergoing testing or still being designed. Like AFATDS, each system was being designed in a “stovepiped” fashion by sponsors that did not have a military-wide view. In the face of this confusion, the Marine AFATDS ORD writers stated:

AFATDS must interoperate with all MAGTF Command, Control, Communications, Computers, and Intelligence tactical data systems. These systems include, but are not limited to the Intelligence Analysis System, Improved Direct Air Support Center, Tactical Combat Operations, Position Location Reporting System, Target Location Designation Hand-off System, and the Contingency Theater Automated Planning System. [AORD00, 2]

Two of these systems have been superseded (Tactical Combat Operations and the Position Location Reporting System), and one has never been deployed (the Target Location Designation Handoff System.) The ORD covers a lot of systems and a lot of ground, and uses the buzzword “interoperate,” but what “interoperate” means is never defined here or in any other requirements documents. As far as USMC systems were concerned, the AFATDS designers were left with no systems with which to test interoperability and no proper guidance on what interoperability meant.

2. Interface Control Documents

Despite the vague ORD, the military acquisition system has a documentation system in place for interfaces. In accordance with this system, the AFATDS program office requires the developer to publish Interface Control Documents, called ICD's. There is one general ICD volume that defines the methodology of AFATDS interface design, and then an ICD is published for each major system that AFATDS must interface with. The systems for which the ICD's are written are listed in the System Segment Specification paragraph 3.2.3. The System Segment Specification lists no fewer than 13 separate interface protocols that AFATDS must comply with, and no fewer than 29 separate systems that AFATDS must interface with. The Intelligence-Operations System (IOS) is not listed, although both JMCIS and GCCS are listed. The GCCS mentioned in the ICD is GCCS -Army, while JMCIS has transitioned to GCCS-M.

The ICD's list the type of messages that must be transmitted between systems, and outlines out what will happen with each message in terms of needed operator intervention to add information or ensure information is not lost. Now imagine that a particular OPFAC has thirty messages arrive an hour – a fairly low rate. If an operator is required to monitor and add information to each message, the chances for error are high. ICD's are an important item for interoperability, but any processing requiring manual steps is unacceptable.

3. DII-COE Compliance Requirements

AFATDS is certified by the Army⁶ to be at DII COE compliance level 6, with waivers for some items, and some items having met a slightly higher compliance level. Recall from the discussion of the DII COE compliance chart that true compliance is not achieved until level 7, with full compliance being level 8. Raytheon is working feverishly to become fully DII COE compliant, and in fact plans on full compliance by the release of GCCS Version 4.x, which was originally expected in late 1999, but hasn't

⁶ As with the other services and their systems, the Army has been delegated the authority to certify its own system by JITC. This is an automatic conflict of interest. [WATK02]

yet been released. Further, AFATDS has implemented some joint segment software that improves the COP. Specifically, AFATDS has added NIMA's Joint Mapping Toolkit (JMTK) to version A99, released this year to the Marine Corps. These efforts drive a continuing need for AFATDS program funding.

D. AFATDS IMPLEMENTATION ISSUES

1. The Master Unit List

The Use Cases listed above gave some indication of the importance of the Master Unit List. The MUL contains the base data upon which all AFATDS operations depend. The MUL is a listing of all friendly units in a theater. The MUL can contain up to 32766 units. If a unit is not listed in the MUL, AFATDS cannot process information to or from that unit. The MUL must be the same between all AFATDS workstations at all OPFACS on the active network. Information stored in the MUL is different than that stored in a friendly unit, but the two are tied together when friendly unit information is entered.

In AFATDS, a target is not an enemy unit. A target is any point on the ground along with a detailed description of what is there to be hit. In fact, the fire support community may target hilltops and road junctions "just in case" there is an enemy force there later. In automated mode, AFATDS chooses whether to shoot such targets based on the Commander's Target Selection Criteria. Also, an enemy unit is not necessarily a target. These targeting and enemy concepts are often at odds with objects of the same name in other C2 systems. Again, semantic confusion is introduced into the interoperability equation.

AFATDS Version A98 data is stored in a relational database called InterBase, from Borland International. ([GCNS95] & [FSSS00, 18]). However, AFATDS is primarily developed by the U.S. Army, who several years ago mandated that all its systems would use the "Joint Common Database" (JCDB). The JCDB is an Informix database designed by the Army to support their Army Battlefield Command System, a collection of C2 systems. The JCDB is used by the Army (principally in GCCS-A), and

it is neither joint across services, nor is it used by all Army systems. However, AFATDS is required to exchange certain classes of information with the JCDB. AFATDS Version A99 implements the JCDB.

2. Current Capabilities

AFATDS does what it was designed to do. It is particularly suited for the Marine Division and below. The artillerymen in the artillery regiment and below have become duty experts. Here, fire mission processing is a daily requirement and the AFATDS operators and support staff have the most familiarity with this complex system. However, above the division level there are major concerns with AFATDS. These concerns stem from the inability of AFATDS to interface with other command and control systems. In particular, MEF staffs are concerned that the one system that “makes decisions” (AFATDS) does not contain the same data as their other Command and Control Systems, which display the Common Operational Picture. Any shadow of doubt about correct data in the AFATDS COP will put AFATDS mission-processing into manual mode, thereby negating the benefits of automating fires in the first place.

Finally, the technical fire control modules of AFATDS (fielded in version A99) have just been fielded to the Artillery batteries, and the early reports are not yet in about this critical task.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. INTEROPERABILITY REQUIREMENTS ANALYSIS

A. INTRODUCTION

The goal of all Command and Control Systems is to build situational awareness. AFATDS and IOS build situational awareness when they present the same data to the commanders in ways that compliment his understanding of the battlefield. Conversely, the more the AFATDS and IOS battlefield informational representations diverge, the greater the loss of situational awareness. The battlefield is complicated enough without the negative effects of having different systems presenting different information. This chapter analyzes two primary operator tasks that require the interoperability of these two systems.

Solving interoperability issues requires system designers to decide “who owns the data.” “Owning the data” means the system in question is the authoritative source for that type of data. In the AFATDS/IOS interface, this question is important because both systems do different things with the data. AFATDS uses high fidelity data to automatically process targets and shoot, while IOS data is of lower fidelity but is more widely shared across the organization. A logical division of data would be for AFATDS to “own” fires data, while IOS “owns” COP data. If this delineation is articulated early, it helps the engineers correctly apportion responsibilities and develop communications that support the decision.

When two systems become interoperable, collaboratively working to meet the same goals, the systems are federated. The collection of collaborating systems is known as a federation. There are three ways that fielded systems can become federated. Either one or both systems hardware can change, the software can change, or the organization can develop workarounds. Generally, the organization will develop workarounds on the spot to resolve system differences. These changes to techniques and procedures are often dangerous in practice because they are ad-hoc and don’t affect the underlying incompatibilities. The advantage however, is that the organization has complete control of their workarounds.

Software has the advantage of being malleable, and is often preferred over hardware integration. Software integration options are to change one or both systems software interfaces, or add middleware that serves as a translator between the systems. The addition of middleware can be either in the form of “wrappers” that surround the target system interfaces, or separate system translators that convert data in between systems. Changing system interfaces means modifying legacy code, which is expertise-intensive and expensive. Adding middleware (such as wrappers and translators) is often seen as easier but has added problems in getting the translation correct and added overhead in terms of time to make the various translations. Significant work has been done in this area by NPS students. Captain Paul Young, USN, has developed the Federation of Independent Objects Model (FIOM) as a part of his PhD dissertation. The FIOM implements translator middleware using XML, XSLT, and automated methods to integrate legacy systems. [YOUN02]

B. LISI-IMM

When evaluating the proper course of action to take in integrating systems, it is useful to state the degree of interoperability that exists between two systems and the degree of interoperability desired. The Department of Defense has adopted the Levels of Information System Interoperability – Interoperability Maturity Model (LISI-IMM) [LISI98] to define inter-system interoperability. The advantage of using this formal interoperability evaluation method is that it evaluates all areas of interoperability in terms of specific and agreed upon interoperability goals.

The LISI-IMM evaluates systems interoperability in terms of the four attributes of Procedures, Applications, Infrastructure, and Data (referred to as *PAID*). The following table lists the attributes with their definition, and some examples of the attribute.

Attribute	Definition	Examples
Procedures	Policies, standards, and guidance leading to the development and deployment of the system(s).	<ul style="list-style-type: none"> • Security Policy • Operating Policy • Workarounds
Applications	Software that enables the system to meet its mission.	<ul style="list-style-type: none"> • AFATDS Target Selection Standards module • IOS Joint Mapping Toolkit
Infrastructure	Items that support a physical and logical connection between systems. This includes system services.	<ul style="list-style-type: none"> • Protocol stacks • Network hardware • Operating System Services
Data	Information Processed by the System to meet the needs of the applications. Data includes both semantic and syntactic data.	<ul style="list-style-type: none"> • IOS Track • AFATDS Unit • IOS Overlay • AFATDS Boundary

Table 16. LISI-IMM Attributes (After [LISI98, 2-8]).

There are five levels of interoperability, ranging from the lowest level of zero to the highest of five. The characteristics of each level are listed below: [LISI98, 2-6]

- **Level 0: Isolated Interoperability in a Manual Environment**

Level 0 describes isolated, standalone systems. No direct electronic connection is allowed or is available, so the only interface between these systems is by manual re-keying or via extractable, common media (i.e., disk). Fusion of information, if any, is done off-line by the individual decisionmaker by other automated means.

- **Level 1: Connected Interoperability in a Peer-to-Peer Environment**

Systems that are capable of being linked electronically and providing some form of simple electronic exchanges. These systems have a limited capacity, generally passing homogeneous data types, such as voice, simple “text” e-mail, or fixed graphic files such as GIF or TIFF images between workstations. They allow decision-makers to exchange one-dimensional information but have little capability to fuse information together to support decision-making.

- **Level 2: Functional Interoperability in a Distributed Environment**

Systems reside on local networks that allow data sets to be passed from system to system. They provide for increasingly complex media exchanges. Formal data models (logical and physical) are present. Generally, however, only the logical data model is accepted across programs and each program defines its own physical data model. Data is generally heterogeneous and may contain information from many simple formats fused together, such as an image with an annotated overlay. Decision-makers are able to share fused information between systems or functions.

- **Level 3: Domain-Based Interoperability in an Integrated Environment**

Systems are capable of being connected via wide area networks (WANs) that allow multiple users to access data. Information at this level is shared between independent applications. A domain-based data model is present (logical and physical) that is understood, accepted, and implemented across a functional area or group of organizations that comprises a domain. Using agreed-upon domain data models, systems must now be capable of implementing business rules and processes to facilitate direct database-to-data-base interactions, such as those required to support database replication servers. Individual applications at this level may share central or distributed data repositories. Systems at this level support group collaboration on fused information products. Decision-making is supported by fused information from a localized domain.

- **Level 4: Enterprise-Based Interoperability in a Universal Environment**

Systems are capable of operating using a distributed global information space across multiple domains. Multiple users can access and interact with complex data simultaneously. Data and applications are fully shared and can be distributed throughout this space to support information fusion. Advanced forms of collaboration (the virtual office concept) are possible. Data has a common interpretation regardless of form, and applies across the entire enterprise. The need for redundant, functionally equivalent applications is diminished since applications can be shared as readily as data at this level. Decision-making takes place in the context of, and is facilitated by, enterprise-wide information found in this global information space.

The AFATDS/IOS interface is at LISI-IMM level two. They do not share the domain-based data model required for systems to be judged at level 3. Level 2 interoperability is not acceptable. The user requirements drive the need for the shared data model given at level 3.

C. USER REQUIREMENTS

1. Requirements

There are two critical requirements. The first is for both systems to provide the same COP. The Marine Corps wants both systems to provide the same COP, so that if one system shows a unit, the other system knows about the same unit at the same place and at the same time. (Recalling the discussion of a view, the second system may not display the same unit, but must know about it.)

The second requirement is to conduct fire missions. The Marine Corps wants to use the IOS (and IOS clients such as C2PC) to Conduct Fire Missions via AFATDS, and get the same results as if the IOS was an AFATDS observer. This includes updating the IOS COP to show the effects of the mission.

2. Actors

The Actors required for the interoperability Use Cases are the union of the Actors required for each individual system. When analyzing the missions of each system along with the Actors, it becomes clear that the differences between the actors conform to real-world differences in missions. So, one cannot make the “Fires Coordinator” actor a member of staff, because the Fires Coordinator has a critical role in preventing fratricide, and that role is differentiated from that of Staff. The following table lists all the actors in the interoperability effort.

Actor Role Name	Description	Example Instances
Commander	The Commander has the same role in both IOS and AFATDS.	<ul style="list-style-type: none"> • Battalion CO • Regimental CO • Division CO
Staff	A member of the commander’s staff fills the staff role. The staff is responsible for implementing the Commander’s policy. Staff provides human oversight of the COP. At least one member of the Staff serves as the COP Track Correlator (The “TOP COP.”)	<ul style="list-style-type: none"> • Intel Officer • Operations Officer • Logistics Officer
Fires Coordinator	The specific role important to the interoperability of the two systems is the role that checks fire missions against the battlefield geometry to prevent fratricide.	<ul style="list-style-type: none"> • Artillery Battalion Liaison Officer • Infantry Battalion Fire Support Coordinator
Sensor	The sensor provides the sensing function. The sensor can be any human or digital information provider. In an interoperable scenario, the sensor is often a Forward Observer.	<ul style="list-style-type: none"> • Reconnaissance Team • Artillery Forward Observer • Counter-fire Radar

Friendly Unit	Any unit so designated by the Commander, whether he controls them or not. (Adjacent units may not be under control, but still have effects in the federation.)	<ul style="list-style-type: none"> • Infantry Company • Artillery Battalion • Mortar Platoon • Airplane
---------------	--	---

Table 17. IOS/AFATDS Federation Actors.

3. Provide COP

AFATDS and IOS must present the commander views of the same COP. In order to achieve this, there must be a common representation of objects between systems. An example of the utility of sharing the same COP would be the ability of a C2PC operator to enter a fire support coordination measure that was accepted and used by the AFATDS network, removing the requirement for staff to switch to an AFATDS terminal every time they needed to change a fire support graphic. Given the COP Use Cases for each system, and the LISI-IMM, here is the evaluation of the current state of affairs for each *PAID* attribute.

- Procedures

Both systems share the same organizational procedures since they both support the Marine Corps “business rules.” The Business rules are the sum of Marine warfighting doctrine. The COP is developed in accordance with the COP handbook, a statement of the business rules. Differences in operational procedures are attributable to differences in system design, and are not significant. There are no hindrances to interoperability from a procedural standpoint.

- Applications

System applications differ because of the different missions each system has. This heterogeneity in applications is required to support the commander and the functional warfare areas. In terms of the COP, however, there is no reason why both systems can’t share the same applications. This is beginning to occur. For example, the Joint Mapping Toolkit (JMTK), a segment for manipulating mapping products and produced by the National Imagery and Mapping Agency (NIMA) to run on DII COE

platforms, has been added to AFATDS A99 software. IOS has had the JMTK since its inception. This commonality across applications is one step toward interoperability.

- Infrastructure

For these systems, infrastructure can be correlated to meeting the requirements of the DII COE model. IOS is level 8 DII COE compliant, while AFATDS is (self-evaluated) level 6 DII COE compliant, with some processes waived. The waivers are given by the service chief for those items that cannot easily be made DII COE compliant. Clearly, AFATDS must become fully DII COE compliant to help ensure interoperability. It must be stressed that DII COE compliance only affects the Infrastructure portion of the interoperability requirement. Further, DII COE compliance may become obsolete in the future, while the requirement for common infrastructure remains. Common infrastructure must be mandated from organizations outside and above the system developer.

- Data

System data is the big issue. IOS and AFATDS have widely different data models, developed for different but related purposes. Each system models the battlefield differently, and there is no common data representation between them. For example, a friendly firing unit is modeled to great detail in AFATDS in order to meet the goals of ammunition tracking and producing firing data. Meanwhile, the same unit is represented in IOS as a unit track, with a course and speed and other non-actionable data. There is no reliable automated way to convert Tracks to Units and vice versa.

Another question is what should be shared. At a minimum, the IOS and AFATDS must have a method of identifying the same object to each other when the two systems communicate. If a friendly firing unit is sent from the IOS to AFATDS, human intervention is required to enter all the data that the IOS doesn't contain that the AFATDS needs to properly process missions. The Track/Unit issue is the example, but most other objects have the same or more complicated issues. The common, shared data model is the primary obstacle to federation COP interoperability.

4. Use Case Conduct Fire Mission

The second critical requirement for Marines is to be able to use an IOS client to call for fire. This presents some interesting organizational issues. Primary among them is authenticating the IOS client as having the authority to initiate fires. For instance, one doesn't want someone thirty miles from the front calling for fire when he has no way of controlling the effects. Generally, only people (or sensors) that can observe the effects of the fires are permitted to call for fire. Also, there are most always area restrictions – a Forward Observer is expected to call for fire in an assigned zone, for example. Yet these are organizational issues for which there are organizational answers that the Marine Corps can work out. The point is a C2PC cannot be used to call for fire right now (without help from middleware)⁷. The following Use Case and sequence diagram represents a proposed usage of the federation.

USE CASE:	Conduct Fire Mission with IOS Client	
Actors:	Observer, Fires Coordinator, Firing Unit	
Purpose:	Deliver fires on targets.	
Overview:	This Use Case describes the steps necessary for AFATDS to deliver fires on targets of opportunity with an IOS Client acting as the observer.	
Type:	Interoperability	
Cross References:	R1.1-1,3, R2.1-2.3 R5.1-5.3, R6.1-6.7, R7.1-7.2	
Actor Action		System Response
1. This Use Case is initiated when an IOS Client (observer) calls for fire. The IOS sends it to the AFATDS in the OPFAC (which is the COC), and the IOS. The message includes the address of the IOS.		
		2. The IOS server identifies the message as a request for fire and records the fire Mission grid for COP purposes. The IOS waits for a target number from the AFATDS. Until a target number is assigned, correlation of the IOS target and AFATDS target is by grid coordinate and sender.

⁷ Middleware (The Fire Support Client) has been developed to do part of this. It is covered in the next chapter.

	3. The AFATDS OPFAC verifies validity of the Call for Fire Grid and other data. AFATDS assigns a target number to the target and sends it to the IOS Client and IOS. AFATDS compares the suspect target to target selection standards and determines attack precedence. AFATDS filters the target, checking for target duplication. AFATDS routes the Call for Fire to a Fires Coordinator based on target location and other properties.
	4. Upon receipt of the target number from AFATDS, the IOS client and IOS update the target.
5. The Fires Coordinator checks the target location against known friendly locations and other parameters. The Fires Coordinator AFATDS system approves or denies the Call for Fire.	
	6. AFATDS determines which unit can range the target, has the appropriate ammunition, and is available for missions. AFATDS sends the mission to that Firing Unit. AFATDS sends a “message to observer” to the IOS Client and IOS telling the Observer what to expect from which Firing Unit.
7. The Observer acknowledges receipt of the message to observer.	
	8. The IOS updates the target with the AFATDS-assigned target number.
9. The Firing Unit acknowledges receipt of the mission.	
	10. The Firing Unit AFATDS processes the Fire Mission by conducting technical Fire Direction and sending the appropriate commands to the gunline.
11. The observer observes the effects on the target and ends the mission. The observer sends a message containing tactical intelligence about the effects on the target to AFATDS and IOS.	

	12. AFATDS sends an End of Mission message to the Fires Coordinator and the Firing Unit. AFATDS saves the tactical intelligence from the mission and updates its databases. AFATDS updates the Firing Unit information by decrementing the amount of ordnance expended from the Firing Unit.
	13. IOS updates the target as “fired” in its COP.
<p>ALTERNATE Courses:</p> <p>Step 3: This step can be done automatically or manually. Current USMC practice is to have a “man in the loop,” so this check is shown as an Actor action, with communication via the AFATDS system.</p> <p>Step 4: If the Mission is Denied, AFATDS sends a Message to observer stating that, and saves the target for later processing or analysis.</p> <p>Step 11: The Observer can choose to adjust the fall of shot based on demonstrated lack of effect on the target, or can repeat the mission if more effects are needed. In these cases, processing continues at step 5.</p>	

Table 18. IOS Client Call for Fire Use Case

In general, mission processing with the IOS Client is very similar to that of processing as if the client was a regular observer. The addition is in the multi-cast capability from the client and AFATDS (in the FSCCRole) to the IOS server, which maintains the updated COP. Multicast is a simple switch from unicast mode, but requires human oversight to set up the dependencies between AFATDS and IOS. The IOS is responsible for identifying the Target state (suspect to active to shot to end of mission) and correlating the target with the relevant IOS client and AFATDS messages. The simple solution of using the target number, assigned by AFATDS, keeps this important responsibility with AFATDS and requires few changes. The following Figure shows the implementation of this Use Case in terms of a UML Sequence Diagram. For the IOS, the COP contains the target; therefore the messages indicate processing between the incoming messages and the COP. Finally, do not read too much into the timing aspects of the apparent parallel processing between the AFATDS nodes and the IOS. Although the processing is time-constrained, the timing is likely to be in terms of seconds rather than milliseconds. For example, the “End of Mission” message from the IOS Client may reach both systems at once but take longer to work through the AFATDS network

because it needs to reach three more nodes. This transmission time will produce some small discrepancies between the systems.

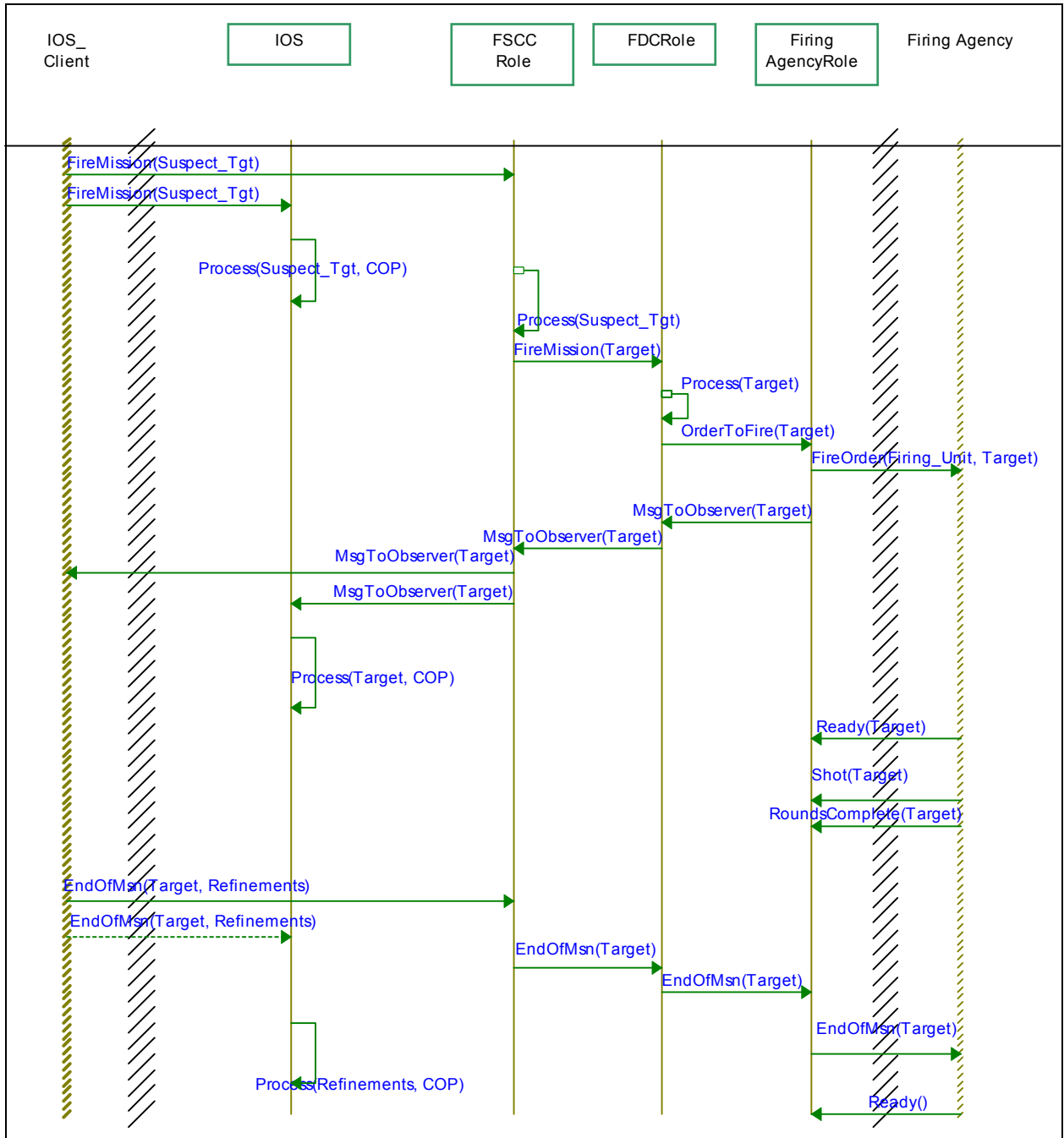


Figure 23. IOS Client Call For Fire Sequence Diagram.

Comparing Figure 23 above to Figure 22 (The AFATDS-only Networked Call for Fire), the Federation is now inside the system boundary, as they are acting as one entity, and should so appear to the end user. The second item of note is the addition of the COP, which is the point of keeping the IOS informed of the mission. The IOS has the responsibility for the COP. How the AFATDS links to the IOS for other COP updates must be developed. Third, note that only the AFATDS FSCC role has any additional duties, and those duties are to add one address (the IOS server) to its mission broadcast tables.

D. THE CRITICAL INTEGRATION CHALLENGE

We return again to the LISI-IMM attributes of *PAID*. There are few problems surmounting issues with *Procedures* because the procedures underlying both systems are the common Marine Corps tactical processes. The *Applications* are what we're trying to integrate, but that becomes easier as the *Infrastructure* evolves to the DII COE standard and common communications (such as the Common Message Parser) for both systems. Instead, the critical issue preventing interoperability is the *Data*. This section discusses specific issues with the data representations that must be fixed.

1. The AFATDS MUL

AFATDS friendly units are limited to what is in the Master Unit List (MUL). Although there can be up to 32766 friendly units, the name of the unit must agree in the MUL across the entire AFATDS network, or else the MUL must be changed to reflect the new friendly unit structure. The MUL includes observers, limiting them as well. The primary reason for using the MUL was to set up the AFATDS database prior to execution to simplify networking parameter entry for the operators. Since AFATDS can use the MUL to figure out what communications protocol a node uses, it doesn't need the operator to set that up. The decision to use the MUL has resulted in a non-scalable and brittle AFATDS network.

2. Friendly Firing Unit Versus Track

The AFATDS friendly firing unit is modeled with a rich data structure in order to accurately monitor and plan fires. Because of its history, the IOS stores information about a unit as a unit track, which does not contain the same amount of data. The operator who creates a friendly unit on an IOS Client must create the same unit in AFATDS, or import the unit from IOS into AFATDS and add the extra data in AFATDS. This leads to dangerous, possibly erroneous duplication and wasted time. This workaround quickly breaks down for any non-trivial situation with numerous units.

3. Unit Identification

There is no common naming standard for track objects and AFATDS units. Therefore, the IOS designers have decided to use the Unit Short Name as the Key field between units. This generates confusion, and every unit has its own Standard Operating Procedure (SOP) for short names. For instance, 5th Battalion 10th Marines (an artillery unit) might be labeled as 5/10. But it could also be 5-10 or 5BN10THMAR. When such a unit hits the IOS/AFATDS interface, an operator must resolve the various differences. There are a certain percentage of tracks that will get by an operator, contributing to COP degradation.

4. COP Object Translation

In AFATDS, the Fire Support Coordination Measures (FSCM's) are called geometries. They are expressed as a collection of up to 30 grid locations and then several other identifying features. The corresponding data in IOS is a polyline in an overlay. The AFATDS designers made a design decision to limit FSCM geometries to thirty locations or less. They made this decision because the current message set will only support transmission of that number of points. This limitation is not present in the IOS polyline. In higher level COC's, such as at the Division or MEF, FSCM's often have at least one hundred grid locations. Therefore, AFATDS FSCM's don't scale in the larger COC's. Further, AFATDS checks these boundaries during mission processing, so a lack

of accuracy translates directly to possible fratricide and the inability to hit certain areas without manual intervention. When FSCM's are translated, an operator is again required to interpret its coordinates on a map.

VII. CURRENT INITIATIVES

The Marine Corps is aware of the issues surrounding the interoperability challenge. There are several initiatives currently in process that attempt to ameliorate the interoperability problems. This chapter lists some of those initiatives and how they will help.

A. FIRE SUPPORT CLIENT

The Fire Support Client is middleware written by Raytheon that allows an IOS Client (such as C2PC) to enter fires data on the Client and send that data to the AFATDS network. Specifically, the client enables the C2PC user to:

- Create and Delete AFATDS Geometries such as Fire Support Coordination Measures
- Receive, filter, and display AFATDS information
- Receive reports on unit status as stored in the AFATDS network
- Creation of targeting lists for processing by AFATDS and reports from AFATDS target lists
- Update unit locations in AFATDS database
- Call for Fire to the AFATDS network as if the Client was a traditional AFATDS observer [FSCA01]

The Fire Support client is effective in extending the AFATDS network to those organizations that do not have a dedicated AFATDS box. Yet the Fire Support Client doesn't solve the interoperability issue because no data in the Client interacts with the IOS data and COP. The user of the IOS client can overlay AFATDS data on his screen, but has no automated way of resolving differences between the objects represented. As an example, imagine a unit represented in each system. If there is a 100-meter grid difference in their locations, and the C2PC operator overlays the AFATDS data on the IOS picture. Then the C2PC operator will see two symbols on his screen. A more dangerous example is if the grids are not close – then the operator is not sure where the unit is actually located, and the difference may be operationally significant.

B. PROXY SERVER

The proxy server is middleware. The AFATDS project main sponsor is the Army. The Army required AFATDS version A99 to migrate to version 4.x data structures in 2000. The GCCS-M interface, known to the Army as the JMCIS interface (which is the same interface that TCO uses, which runs on an IOS box) used UB version 3.x. The GCCS 4.x and UB v 3.x are incompatible, so Raytheon wrote a proxy server that would reside on JMCIS/GCCS-M/TCO/IOS boxes (as a segment on the Version 3.x UB) that would translate v4.x data into v3.x data. The proxy came in because the segment riding on the UB v3.x used the old API's. [APRX00] This is an example of middleware being used to fix a versioning issue.

The proxy server is effective in translating those items that can be translated between IOS and AFATDS. A system operator is still required to enrich the data model where the data does not exist and ensure that short titles of the units are properly translated between the systems. Finally, as GCCS transitions to version 4.x, the proxy server should be replaced by the communications methods common to the infrastructure underlying both systems.

C. TESTING

The Marine Corps Tactical Systems Support Activity (MCTSSA is the west-coast part of SYSCOM) has the charter to conduct functional (or "black box") testing of recently released software for compatibility. They have built a facility that has all currently fielded Command and Control hardware and software, connected by fielded tactical communications devices. In addition, they have some ability to inject software faults and hardware failures into testing to evaluate the results. MCTSSA refers to their test suite as "The Node."

I observed the MCTSSA test of the AFATDS/IOS connection in response to a Safety-of Use Message sent from the I MEF current fires officer in January 2002. The message alleged several critical safety deficiencies in the interface that could result in fratricide. [1MEF02] The AFATDS project officer recommended not using the AFATDS/IOS interface until testing could be completed at MCTSSA. The interface was

tested in late February 2002 and was indeed found to be inoperable. Specifically, no COP objects could be shared between the AFATDS and IOS Networks (both networks still worked internally) despite the proxy server. Knowing this information, the AFATDS project officer directed that the new software versions of both systems be tested against each other. These versions did interoperate at the previous level (LISI-IMM level 2), and the MARCORSYSCOM made the logical decision to rapidly issue the new software to the Fleet Marine Forces.

What is interesting is that neither MCTSSA nor MARCORSYSCOM knew that the software interface didn't work until the FMF reported it. From a MCTSSA perspective, there are thirty-plus C2 programs out there, and each one changes versions on its own schedule, usually annually – therefore the testing problem is intractable. From the FMF perspective, MCTSSA and MARCORSYSCOM appear to be reactive vice proactive.

MCTSSA (Systems Engineering and Integration Branch) does have one really outstanding testing project called the MAGTF C4SIR Integrated Program (MIP). The MIP is attempting to baseline⁸ all the C2 systems in the Corps and test them in a federation rather than individually. Testing is done using an “end-to-end” concept, sending messages in a tactical environment and determining total transmission time. This method is effective because it highlights federation failures rather than individual system failures. For instance, the MIP program may measure the amount of time it takes to send a Call for Fire from an observer to the Firing battery. Perhaps the AFATDS program has promised processing times in terms of milliseconds, but the end-to-end test indicates a time approaching two minutes – then the MCTSSA engineers can troubleshoot the delay. The fact that the MIP is baselined allows for scientific analysis. MARCORSYSCOM is suspicious of the MIP program because the baseline constrains Project Officers from releasing products whenever their individual programs are ready. The Fleet may not accept baselining because individual communities may feel they don't have “the latest software.”

⁸ Baselining means that individual systems are under version control, and software system versions are not issued to the fleet without being tested and put into the new baseline, which would coordinate Corps-wide release of all software in the new baseline.

D. FLEET SUPPORT CONTRACTORS

Each company that provides a C2 system also has fleet support contractors on the payroll that work at the major commands such as the MEF's. These contractors serve as liaisons between the manufacturers and the units, report bugs, implement fixes, and act as "go to guys" for their systems. The fleet support contractors are experts in their systems, and they become expert in the needs of the fleet. They serve as advocates for improved systems and can better articulate requirements.

However, there are issues with civilians in a military organization. First, the C2 systems have requirements for ease of use, and in fact, these requirements are the ones that are properly written in the ORD's. Needing a contractor means the system did not meet usability requirements. Second, every contractor that must be hauled to a battlefield takes the place of a combat-ready Marine. Third, there are obvious safety and liability issues with civilians on the battlefield. Fourth, the contractor works for the company that makes the C2 system. He just happens to be posted with the Marine Corps. Of course they will support the Corps, but their viewpoint is that of their system, rather than federation interoperability.

E. VERSION CONTROL

The SE&I branch of MARCORSYSCOM is attempting software version control. They have made several efforts at coordinated version control between the contractors, and tied this effort into the MIP baseline effort at MCTSSA. The current system is called "C4I for the Warrior" (C4IFTW) and is supported by a password-protected database called MSTAR. MSTAR stands for the MAGTF Systems/Technical Architecture and Repository. [MSTR02] The MSTAR's mission is to:

...provide a MAGTF C4ISR system-of-systems that is controlled, secure and interoperable in the Joint Environment, from the battle field to the sustaining base. MSTAR is a suite of web accessible products that provide Marine Corps acquisition professionals information they require to ensure systems interoperability. [MSTR02]

MSTAR should contain all the I-KPP's for each MARCORSYSCOM project, along with the development and deployment data for each system. However the database is empty when queried for AFATDS (both version A98 and A99), and calls and emails to the listed points of contact were not returned. Therefore it is impossible for me to determine if this method of data collection and coordination is working or a failure.

But even if SE&I branch had a perfect tracking mechanism for every C2 project in the Corps, the acquisition culture rewards project officers that meet cost, schedule, and performance constraints. They are not rewarded for spending time with SE&I division talking about interoperability. Because project officers are pulled from the fleet (as was the case with the last two AFATDS project officers) there is no institutional memory, and each officer focuses first on the budget. The senior leader in the ground C4 project office has control of 11 different C2 programs but no interoperability tasking. He has never been provided the specific role of software integrator, and doesn't have the training or the expertise to coordinate version control between all the various systems. The bottom line is there is no penalty for not meeting version or interoperability goals.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. CONCLUSIONS

A. GENERAL

The AFATDS/IOS interface is at LISI-IMM level 2, which permits limited interoperability with significant user intervention for data conversion between the systems. The current initiatives to apply third party middleware to solve the problem are only partially successful. Further, the middleware solutions are only effective until either software version changes, which occurs annually at a minimum, although through luck a middleware solution may make it through a version change. The goal of a truly interoperable system is to reach LISI-IMM interoperability level 4. In order to reach this goal, the AFATDS/IOS interface must overcome the hurdle of different data models. The major hinderances to reaching interoperability level 4 are organizational and political rather than technical.

Even a cursory search of the relevant literature reveals that technical solutions to interoperability abound. A good interoperability solution will be open source, shared across many domains, and focus on integrating the system-of-systems data models. Possible technical solutions include ideas like Captain Young's Federation of Independent Objects interoperability Model (FIOM), the use of Extensible Markup Language (XML), or similar languages and structures that support information sharing. No one of these technologies completely solve the interoperability problem, but they do provide a good foundation for further work. The challenge of moving an adequate number of bits through the air is being met, and the protocols necessary for low-level communication between systems is no longer a challenge. Now it is time to focus on the data.

No matter what solution or combination of solutions, each will require the expenditure of significant sums of money and time. Solving interoperability issues will also require expertise and a degree of cooperation from every DoD agency heretofore unseen. Yet, the need is pressing. Every officer will run up against an interoperability challenge sometime in his or her career. Tactical interoperability problems increase the potential for casualties. Rejecting automated Command and Control systems or ignoring

them in the hopes they will go away is a futile strategy. Instead, there are specific small steps that can be taken by each level of the hierarchy to solve interoperability issues. They are listed below.

B. JOINT RECOMMENDATIONS

The JROC must force interoperability. They have been given the authority, and have been tasked with the responsibility for integration. They must not settle for service appeasement. In order to integrate, the JROC must rely on Subject Matter Experts. DISA has the expertise. DISA already has a dedicated and influential role, but the services must follow DISA guidelines in order to solve interoperability problems. Services are reluctant to do this not only because of service parochialism, but because they bear the pain of cost overruns and schedule delays associated with changing requirements. Yet without JROC control, the question of “who’s in charge” will continue to paralyze the joint world.

The DoD must agree on a joint standard for naming objects in the tactical environment. The most basic subset of this problem is to agree on a Unit Reference Number (URN) standard. The URN would be the key, unchanging attribute allowing conversion of real-world units between the battlespace and the various systems. Currently, service parochialism is preventing the DoD from agreeing on a URN standard.

C. MARINE CORPS RECOMMENDATIONS

1. System Architecture Management

The Marine Corps suffers because of a lack of an overall systems architecture that supports the operational architecture. It further suffers because the organizations tasked with making that architecture do not have the power necessary to make it a reality. In accordance with the Clinger-Cohen Act of 1996, the Marine Corps appointed a Chief Information Officer (CIO), tasked by statute with managing all enterprise-wide Information Technology systems. From the title and responsibilities assigned, one would think that architecture would be the CIO’s bread and butter, and in fact the CIO has had

extensive involvement in IT decisions on the non-tactical side (this includes the Navy-Marine Corps Intranet effort). But for political and other reasons, the same has not happened in the tactical arena.

The CIO must take charge and assume responsibility for the tactical Systems Architecture effort. A good delineation of effort would be achieved if the CIO dictated the Systems Architecture in accordance with the Operational Architecture generated out of the Marine Corps Combat Development Command. Then the Marine Corps Systems Command could buy or build systems to the Systems Architecture, using the already established and accepted Joint Technical Architecture.

The Marine Corps needs to articulate its concerns with the Unit Reference Number (URN), and then submit to a joint decision and implement that decision Corps-wide. The choice of a number to represent units should take less than a year. Any more time indicates political inertia is hurting interoperability.

2. MARCORSYSCOM

Until the CIO assumes responsibility for Systems Architecture, SE&I Branch, MARCORSYSCOM, must be given the authority to halt and re-engineer any given program to meet interoperability requirements. There is no doubt that requiring Project Officers to meet interoperability requirements is culturally painful and institutionally costly, but the Marine Corps must make its procurement people face this pain rather than fielding non-interoperable systems, where the FMF suffers. This is not happening, as indicated by the lack of usable data in the MSTAR database, the current SE&I effort to this end.

The IOS program should use formal program management methods and have its own funding line in accordance with DoD Acquisition regulations. This will allow better management of the software and creation of other artifacts, and most importantly, accurately articulate and document user requirements. Since IOS is a “son of GCCS”, the IOS program has the larger task of ensuring USMC requirements are met in the parent software. The IOS program manager and engineers should not be software writers but requirements articulators.

The AFATDS project office must continue to fund short-term solutions that enable LISI-IMM level 2 interoperability. This includes both the Fire Support Client and the Proxy Server. Decision-makers must be made aware that these software programs must have continuing funding and support, because they will require modification at roughly twice the rate of the supported programs. Interface testing will also require funding at current rates, which supports an AFATDS project officer at MCTSSA.

3. Fleet Contractors

Because current and legacy systems are not user friendly and do not interoperate, contractors (or TechReps) in the FMF are required for the foreseeable future. However, the FMF should stop hiring program contractors and should start hiring “Interoperability Engineers.” These interoperability engineers, hired from consulting firms or from other sources, would work for the FMF, rather than for any particular system, and be evaluated on their ability to achieve systems interoperability. The Interoperability Engineer skillset could be met with a bachelor’s degree in Computer Science or a related field, and must include the ability to accurately articulate and document user requirements.

4. MCTSSA

The Marine Corps (possibly MCTSSA) should implement a collaborative website for users, administrators, and developers to share information. It should be unclassified but password protected. Keeping the website on classified networks impedes people’s ability to use the information. One outstanding website model is SPAWAR Charleston’s TACMobile website for ground C2 systems. There is unlimited shelf space for training materials and user’s manuals. Chat and other collaborative tools are built in. Such a website would require at least one moderator, but would bear great dividends in increased information sharing between users and developers. MCTSSA already maintains a 24 hour voice helpdesk and a classified website, which is a start.

MCTSSA’s end-to-end testing (the MIP) must continue. The MIP process should set baselines and formally report to the CIO at least annually. If there is a political issue with MCTSSA being under MARCORSYSCOM and not under the USMC CIO, then the

MIP process should have the appropriate blessing by the appropriate Marines to become a CIO-directed effort. End-to-end testing is the only way to accurately assess and quantify the warfighter's complaints. Accurate problem descriptions will lead to better solutions.

D. RECOMMENDATIONS FOR NPS

The Naval Postgraduate School is the Navy and Marine Corps' corporate institution for graduate technical education. This year alone, some 20 Marines are graduating from the ITM curriculum. Upon departing NPS, every Information Technology and Software Engineering officer will be expected to do their part in solving the myriad interoperability problems in the fleet. Yet the NPS curriculum has no course that discusses interoperability issues or interoperability solutions in any depth. Such a course should be developed. For Software Engineering students, an interoperability course should be required. Possible topics for an interoperability course include:

- Problems with Software integration in DoD
- System-of-System modeling and Federations
- LISI-IMM and other methods of evaluating current interoperability issues
- Generating Interoperability Requirements
- Shared Data Environments
- Evaluating Middleware and Glue and Wrapper Software
- Data conversion using XML and XSLT
- The Promise and Pitfalls of Object Orientation in integration efforts

The ultimate goal of the course would be to prepare students to evaluate third party vendor solutions to interoperability challenges, since a common technique for interoperability is to buy middleware or convert data from one system to another. Another distressingly common technique is to generate manual workarounds or ignore the problem. Graduates having the knowledge from a course similar to this would be much better equipped to handle the inevitable interoperability challenges that they will face.

E. THE UML AS A MODELING TOOL

Important modeling considerations are the amount of coverage of the model (its scope), and the accuracy with which the model represents reality (its fidelity). A critical element of a modeling language is the ability to accurately and unambiguously convey the meaning in the model. The UML is associated with object oriented analysis and design, and meets these critical elements for object oriented models and eases development of the corresponding systems.

The UML was used in this thesis to model two systems in the C2 domain that were not originally designed or implemented using object orientation. Model scope was limited because of domain complexity. Fidelity was limited for the same reason. However, it is clear that the Use Cases and resulting UML were easy to learn, accurately reflected the concepts in the Use Cases, and effectively conveyed the meaning of the systems. Further, the graphical features of UML led to better understanding of the systems and interoperability challenges, and pointed to the complexities of system integration. Although timing and other constraints were not modeled, the UML (and other associated modeling tools such as the Object Constraint Language) has sufficient depth to model these factors.

Finally, object orientation appears to be an appropriate paradigm for C2 systems development. Most battlefield items have attributes and methods that can map directly to software. Thinking about the battlefield in this way may help warfighters better converse with system developers to describe their requirements. Using a common visual language such as UML can only help.

F. AREAS FOR FURTHER RESEARCH

AFATDS and IOS need to share data, and the main sticking point is a shared data model. Research could be directed at establishing that shared data model from conceptual and business case rules. Midterm solutions could focus on development or selection of appropriate middleware products. Another midterm solution that warrants further research is the use of XML and XSLT to develop middleware that automatically


converts data from one system to another. A concrete topic would be to apply Captain Young's FIOM (which uses XML) to the data models in the AFATDS/IOS interface.


Research could be conducted into what Commanders actually want. This could involve prototyping several different COP models and see which is most popular for a given domain. Research should be conducted into future C2 systems. The Marine Corps Warfighting lab, in conjunction with the Collaborative Agent Design Research Center at the California Polytechnic Institute, is developing a shared Command and Control Data Environment. The system using this data environment is called the Integrated Marine Multi-Agent Command and Control System (IMMACCS). [POHL01] Although this initiative is only service-wide (meaning deployment would create a one-service stovepipe), it could bear fruit in helping define the warfighter's need.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- [1MEF02] Sartor, LtCol USMC, G-3 Current Fires, First Marine Expeditionary Force, TRACK LATENCY DEFICIENCIES WITHIN THE ADVANCED FIELD ARTILLERY TACTICAL DATA SYSTEM, Feb 2002.
- [AHL99] Headquarters, U. S. Marine Corps, Operator's Manual, Advanced Field Artillery Tactical Data System (AFATDS) Operational System Software (A98), Vols. 1-4 UM-10690A-10/1-4, 15 October 1999.
- [AMHD85] The American Heritage Dictionary, Houghton-Mifflin, Boston: 1985.
- [AORD00] Marine Corps Systems Command, Marine Corps Operational and Organizational (O&O) Concept for the Advanced Field Artillery Tactical Data System (AFATDS), w/Ch 1, 15 February 2000.
- [APIC02] Marine Corps Tactical Systems Support Activity, Advanced Field Artillery Tactical Data System (AFATDS)/C2PC Fire Support Client (CFSC), January 2002. Online at: http://www.mctssa.usmc.mil/PSD/C4I Apps/AFATDS_CFSC/AFATDS.html
- [APRX00] Mauney, Leslie, Raytheon Systems Engineering, AFATDS Internal Design Approval Memo, "Modify AFATDS Software to use Proxy Server API's for JMCIS Interface", 26 September 2000.
- [BOOC99] Booch, Grady, James Rumbaugh, and Ivar Jacobson, The Unified Modeling Language User Guide, Addison-Wesley: Reading, MA, 1999.
- [BRLA61] Ballistic Research Laboratories, Aberdeen Proving Ground, Report No. 1115, "A Third Survey of Domestic Electronic Digital computing Systems" : 1961. <http://ed-thelen.org/comp-hist/BRL61-f.html>
- [BUDD02] Buddenberg, Rex, Professor, Naval Postgraduate School, Personal interview, papers, and emails, 6 May 2002.
- [BULL02] Bullard, Steve, GCCS-M Software Manager, SPAWAR San Diego, Code 157, Personal interview and email, 2 July 2002.
- [CHBK98] Department of Defense Chief of Staff, Operations Division (J-33), Common Operational Picture Handbook for GCCS 3.02, Version 2, 31 July 1998.

- [DAST92] Dastrup, Boyd L., King of Battle: A branch History of the U.S. Army's Field Artillery, U.S. Army Training and Doctrine Command (TRADOC), 1992.
- [DATL02] Office of the Under Secretary of Defense for Acquisition, Technology & Logistics, Office of the Director, Interoperability. 13 February 2002. Online at: <http://www.acq.osd.mil/io/sa/support.html>.
- [DDIG94] Office of the Inspector General, Department of Defense, Milestone Review Process for the Advanced Field Artillery Tactical Data System, May 27, 1994. 
- [DIMP98] Office of the Assistant Secretary of Defense, Command, Control, Communications, and Intelligence (ASD C3I), Defense Information Infrastructure Master Plan Version 7.0, 11 May 1998.
- [DIRS97] Joint Interoperability and Engineering Organization, Defense Information Infrastructure (DII), Common Operating Environment (COE), Integration and Runtime Specification (I&RTS), Version 3.0, July 1997.
- [DODD02] Department of Defense Directive 4630.5, Interoperability and Supportability of Information Technology (IT) and National Security Systems (NSS), 11 January 2002.
- [FCWK01] Dorobek, Christopher J., "Government Ok's New Info Grid," Federal Computer Week, 1 October 2001. Online at: <http://www.fcw.com/fcw/articles/2001/1001/pol-grid-10-01-01.asp>
- [FSCA01] Marine Corps Systems Command, Statement of Work for the C2PC-AFATDS Fire Support Client, Initial Version, Quantico: 6 July 2001.
- [FSSS00] Raytheon Company, System Segment Specification for the AFATDS V2.1, Revision B, Contract No. DAAB07-90-C-E708, CDRL Sequence No. H509: Fort Wayne, 2000.
- [GCNS95] "Standard Query, Where's My Ammo?," Defense Department Briefs, Government Computer News, 30 October 1995.
- [GSDD01] Space and Naval Warfare Systems Command (SPAWAR) Code 157, Global Command and Control System – Maritime (GCCS-M) Version 3.1.2.1 Segment Description Document (SDD), 16 March 2001.
- [IEEE90] Institute of Electrical and Electronics Engineers, IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries, New York, NY: 1990.

- [JCOP02A] Chairman of the Joint Chiefs of Staff Instruction 3151.01A, Global Command and Control System Common Operational Picture Reporting Requirements, 1 March 2002.
- [JJPS95] Chairman of the Joint Chiefs of Staff, User's guide for JOPES (Joint Operation Planning and Execution System), 1 May 1995.
- [JPUB01] Chairman, Joint Chiefs of Staff, DoD Dictionary of Military and Associated Terms, (Short Title: JP 1-02), 2002.
http://www.dtic.mil/doctrine/jel/new_pubs/jp1_02.pdf
- [KUBI01] Kubicki, Adam, Captain, USMC, AFATDS Project Officer, Marine Corps Systems Command, personal interview, December 2001.
- [LARM98] Larman, Craig, Applying UML and Patterns: an Introduction to Object-Oriented Analysis and Design, Prentice-Hall: Upper Saddle River, NJ, 1998.
- [LEDE99] Lederman, Gordon N., Reorganizing the Joint Chiefs of Staff: The Goldwater-Nichols Act of 1986, Westport: Greenwood Press, 1999.
- [LISI98] Office of the Assistant Secretary of Defense (C3I), C4ISR Architecture Working Group, Levels of Information System Interoperability, 1998. Online at:
http://www.c3i.osd.mil/org/cio/i3/AWG_Digital_Library/index.htm
- [LITT00] Litton Data Systems, "Artillery Fire Control Systems," 2000. Online at: <http://www.littondsd.com/programs/afcs.html>
- [LITT02] Little, Laura, Major, USMC, The Digitized COC: After Action Report from 6th Marines Combined Arms Exercise, February 2002, prepared for the Tactical Training and Exercise Control Group, Marine Corps Air-Ground Combat Center, 29 Palms, CA, February 2002.
- [LOCH02] Locher III, James R., Victory on the Potomac: The Goldwater-Nichols Act Unifies the Pentagon, Texas A & M University Press, College Station: 2002.
- [MARC02] Marcinkowicz, Tom, Major, USMC, 6th Marines Communications Officer, Personal interview, January 18, 2002. 
- [MCDP06] Headquarters, United States Marine Corps, Command and Control (MCDP-6), 4 October 1996.

- [MSEI02B] Pasagian, AJ, Major, USMC, MARCORSYSCOM SE&I Project Officer, SE&I: A Common Sense Approach for Systems Interoperability, presentation, February 2002.
- [MSEI02C] Pasagian, AJ, Major, USMC, MARCORSYSCOM SE&I Project Officer, Near Term Systems Architecture, presentation, September, 2000.
- [MSTR02] Systems Engineering and Integration Division, Marine Corps Systems Command, MAGTF Systems Technical Architecture and Repository (MSTAR), 2002. Online at: <http://www.marcorsyscom.usmc.mil/mstar/mstarsplash.html>. This website is password protected.
- [PECK02] Peck, Eric, Captain, USMC, TCO and IOS Project Officer, Marine Corps Systems Command, interview 12 Jun 2002.
- [POHL01] Pohl, Jens, et. al., IMMACCS: A Multi-Agent Decision Support System, Collaborative Agent Research Design Center, San Luis Obispo, CA, 2001.
- [SNIN01] SECNAVINST 5000.36, Department of the Navy Data Management and Interoperability, 1 Nov 2001.
- [SPAW02] Space and Naval Warfare Systems Center, Charleston, TACMOBILE Website, 2002. Online (password protected) at: <https://tacmobile.spawar.navy.mil/tmvhome/>
- [TCOE95] Marine Corps Combat Development Command, Concept of Employment for the Tactical Combat Operations (TCO) System, 9 May 1995.
- [TIPS95] Commander, Marine Corps Systems Command, Integrated Program Summary for Tactical Combat Operations (TCO), signed by Gen. Mutter, 29 Dec 1995.
- [TMNS92] Marine Corps Combat Development Command, Mission Needs Statement for a Tactical Combat Operations System, 16 June 1992.
- [TORD95] Marine Corps Systems Command, Operational Requirements Document for Tactical Combat Operations (TCO ORD), with changes 1 and 2, 25 April 1995.
- [WALK01] Walker, Robert, DII COE Program Manager, DISA, DII COE Overview: Presentation to the DII COE Technical Exchange, 15 May 2001.
- [WATK02] Watkins, Stan, LtCol, USMC (ret), artillery officer, personal interview, 7 May 2002.

[YOUN02] Young, Paul E., Heterogeneous Software System Interoperability through Computer-Aided Resolution of Modeling Differences, PhD, dissertation, The Naval Postgraduate School: Monterey, CA, June, 2002.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. GLOSSARY

<u>Term</u>	<u>Definition</u>	<u>Reference</u>
ADLER	A German Artillery C2 system. The acronym stands for “The Artillery Data Situation and Deployment Computer Network.”	
AFATDS	Advanced Field Artillery Tactical Data System	
API	Application Program Interface. A standardized way for programmers to call the functions of a given piece of software without needing to know the details of the software code.	
Artifact	“A piece of information that is used or produced by a software development process.” Examples of artifacts include source code, help manuals, and written or electronic documentation.	[BOOC97]
BATES	A British Artillery C2 system. The acronym stands for “The Battlefield Artillery Target Engagement System.”	
bps	Bits per second. A measure of the digital capacity of a communications channel.	
C2	Command and Control	[JPUB01]
C4I	Command, Control, Communications and Computers, and Intelligence	
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance	
CCIR	Commander’s Critical Information Requirements	
CMP	Common Message Processor	
COP	Common Operational Picture: “A single identical display of relevant information shared by more than one command.” A common operational picture facilitates collaborative planning and assists all echelons to achieve situational awareness.	[JPUB01]
COTS	Commercial Off the Shelf. Refers to commercial (unmodified) software bought for government use.	
CROP	Common Relevant Operational Picture: A term representing a portion of the COP, relevant to a given echelon or situation.	
CTP	Common Tactical Picture: A term representing a portion of the COP, relevant to a given echelon or situation.	
DACT	Digital Automated Communications Terminal. This digital device is one of several used by forward observers (FO’s) to call for fire.	
DII-COE	Defense Information Infrastructure Common Operating Environment.	
DISA	Defense Information Systems Agency	

EPLRS	Enhanced Position-Location Reporting System	
FA CP	Field Artillery CP. An Army term equivalent to the USMC's Fire Direction Center (FDC).	
FDC	Fire Direction Center. A generic term representing the C2 node where technical fire direction computations occur (in units with indirect fire platforms.) In higher headquarters, the FDC's are responsible for a mix of technical and tactical fire direction.	
FMF	Fleet Marine Forces. This term refers to Marine combat organizations in a general sense, independent of task organization for specific missions.	
FSCC	Fire Support Coordination Center. The FSCC is a C2 node responsible for conducting tactical fire direction, fire planning, and clearance of fires (a safety function). Typically, FSCC's are part of supported infantry units.	
FSCM	Fire Support Coordinating Measure. "A measure employed by commanders to facilitate the rapid engagement of targets and simultaneously provide safeguards for friendly forces." FSCM's are drawn on maps and contain instructions about what is permitted or denied in an area. An example FSCM is the No Fire Area (NFA).	[JPUB01]
FSE	Fire Support Element. An Army term equivalent to the USMC's Fire Support Coordination Center (FSCC).	
GCCS	Global Command and Control System	
GOTS	Government Off the Shelf. A term referring to software developed under governmental auspices but made available to others for use.	
IAS	Intelligence Analysis System	
IER	Information Exchange Requirement: The requirement for information to be passed between and among forces, organizations, or administrative structures concerning ongoing activities. Information Exchange Requirements identify who exchanges what information with whom, as well as, why the information is necessary and how that information will be used.	[SNIN01]
Integration	The act or process of making into a whole by bringing all the parts together. To join with something else; unite.	[AMHD85]
Interoperability	1. The ability of two or more systems or components to exchange information and to use the information that has been exchanged. 2. The condition achieved among communications-electronics systems or items of communications-electronics equipment when information or services can be exchanged directly and satisfactorily between them and/or their users. The degree of interoperability should be defined when referring to specific cases.	1. [IEEE90] 2. [JPUB01]

IOS	Intelligence-Operations Server
IUC	Independent User Center. In the AFATDS domain, an IUC is a standalone AFATDS workstation with a subset of the capabilities of an OPFAC.
JMCIS	Joint Maritime Command Information System
JOPES	Joint Operations Planning System. This acronym describes a process the output of which is Operations Plans and deployment information.
JROC	Joint Requirements Oversight Council
JTA	Joint Technical Architecture
JVMF	Joint Variable Message Format
MARCORSYSCOM	Marine Corps Systems Command. This command is responsible for development and acquisition of all Marine Corps materiel, to include software. Also referred to as SYSCOM.
MCTSSA	Marine Corps Tactical Systems Support Activity
MEB	Marine Expeditionary Brigade. This combined arms force is built around an infantry regiment ground combat element, a composite air group as the air combat element, and a brigade service support group as the service support element. It is larger than a MEU but smaller than a MEF.
MEF	Marine Expeditionary Force. This largest of the Marine's combined arms forces has a division as the ground combat element, an air wing as the air combat element, and a force service support group as the service support element. It is commanded by a Lieutenant General.
MET	Meteorological Information
MEU	Marine Expeditionary Unit. This combined arms force has a battalion landing team as the ground combat element, a composite squadron as the air combat element, and a MEU service support group as the service support element. It is the smallest sustainable combined arms force.
MIDB	Modernized Integrated Database. This database stores intelligence information.
MUL	Master Unit List. In the AFATDS system, the MUL is a listing of all friendly units that can be joined in an AFATDS network.
OODA	Observe-Orient-Decide-Act
OPLAN	Operations Plan. A collection of documents, maps, and other items that outline the conduct of a particular course of action. It may or may not have any relevance to any particular world situation.
ORD	Operational Requirements Document. In acquisition, a program must have an ORD before it can become a Program of Record.

PLI	Position-Location Information	
PLRS	Position-Location Reporting System	
Program of Record	Generically used to represent a development or Acquisition program under the DoD 5000 Acquisition model. Generally, a Program of Record is one that has a validated Mission Needs Statement and appropriated funding.	
SPAWAR	Space and Naval Warfare Systems Command. This command is responsible for development and acquisition of C2 systems for the U.S. Navy and when tasked, for the U.S. Marine Corps.	
SYSCOM	Marine Corps Systems Command. This command is responsible for development and acquisition of all Marine Corps materiel, to include software.	
Systems Architecture	Design. The way components fit together.	[IEEE90]
TCO	Tactical Combat Operations	
TDBM	Track Database Manager	
TPFDD	Time-Phased Force Deployment Data. A set of documents that describe how forces and equipment will get from one place to another in response to military need.	
Track	A track is a single entity reported on the COP such as an aircraft, ship, TBM or emitter location. A track can also designate an aggregation of military personnel, weapon systems, vehicles, and support elements or any other operationally significant item.	[JCOP02A]
UB	Unified Build. This DISA term represents the common DISA managed software components in a Global Command and Control (GCCS) suite.	
UCP	Unified Campaign Plan	
View	In UML: A projection into a model, which is seen from a given perspective or vantage point and omits entities that are not relevant to this perspective.	[BOOC99, 468]
XML	Extensible Markup Language. This meta-language stores the definition and presentation characteristics of data. By storing this data about data, system developers can specify and manipulate complex data models.	
XSLT	Extensible Style Language Transformation. This language specifies the transformation of data from one XML definition to another.	

APPENDIX B. EXAMPLE MARINE ORGANIZATION FOR COMBAT

Often, Marine artillery Battalion will be in Direct Support of an Infantry Regiment. The term “Direct Support” means that the artillery Battalion sends liaison teams to the Infantry Regiment, and priority for artillery fires goes toward Regimental Needs. The artillery Battalion provides Forward Observer teams to the supported infantry Regiment, and answers calls for fire from its own observers first. Figure 24 below shows this typical organization.

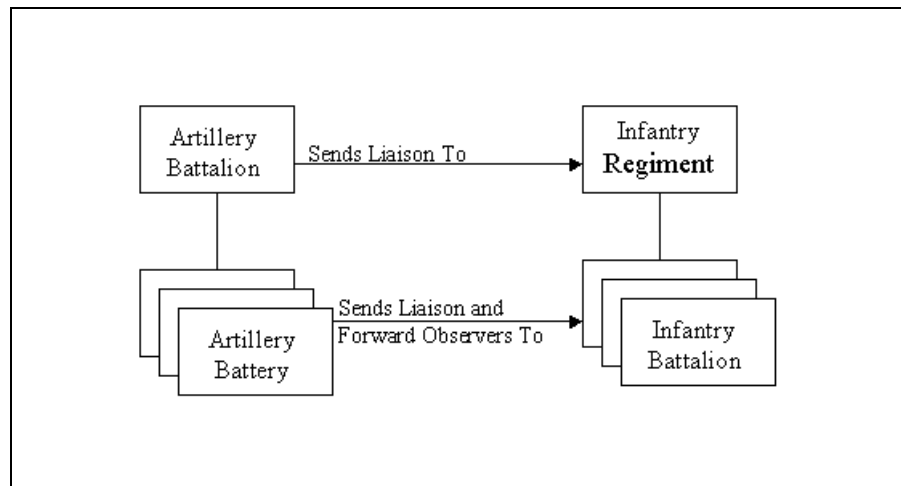


Figure 24. Marine Artillery Battalion in Direct Support of an Infantry Regiment

The Liaison team that attaches to the Regimental headquarters is led by an artillery Captain or Major. Together with his staff and others from the Regiment, he forms the Fire Support Coordination Center (FSCC). The FSCC is responsible for planning and executing all fires in the Regimental zone of action, or battlespace. The Artillery Battalion Liaison cell travels with sufficient AFATDS terminals to support the Regimental Combat Operations Center (COC).

Each of the three Artillery batteries supplies 3 forward observer teams and a Liaison team to an Infantry Battalion. In the case of the infantry battalions, the battery liaison officer joins the Battalion FSCC, which is headed by an infantry officer, usually the Infantry Battalion Weapons Company Commander. The artillery Liaison Officers bring sufficient AFATDS terminals to support the Infantry Battalion COC's. Meanwhile,

the forward observers (a 2nd or 1st Lieutenant along with several other Marines) each join a company and provide fire support expertise as well as radio communications sufficient to communicate with the Artillery Regiment. The forward observers bring sufficient digital equipment to support communications with the AFATDS terminals at the infantry battalion FSCC's.

One typical Digitized Regimental COC is diagrammed in Figure 25. This COC was designed by the 6th Marines Regimental Commander, Col Coleman and his staff. It was used for the 6th Marines Combined Arms Exercise in 29 Palms, Ca., in February and March 2002. The artillery battalion supporting 6th Marines was 2nd Battalion, 10th Marines. Although this diagram shows one AFATDS terminal attached to the COC, the Artillery Battalion typically brings at least two terminals. The second terminal is used for planning. Also note the IOS (v. 1) Server located in the entrance to the main tent, and the Command and Control PC (C2PC) computers, which are IOS client stations. Finally, all the systems were networked via Ethernet. Networking is necessary but not sufficient for interoperability.

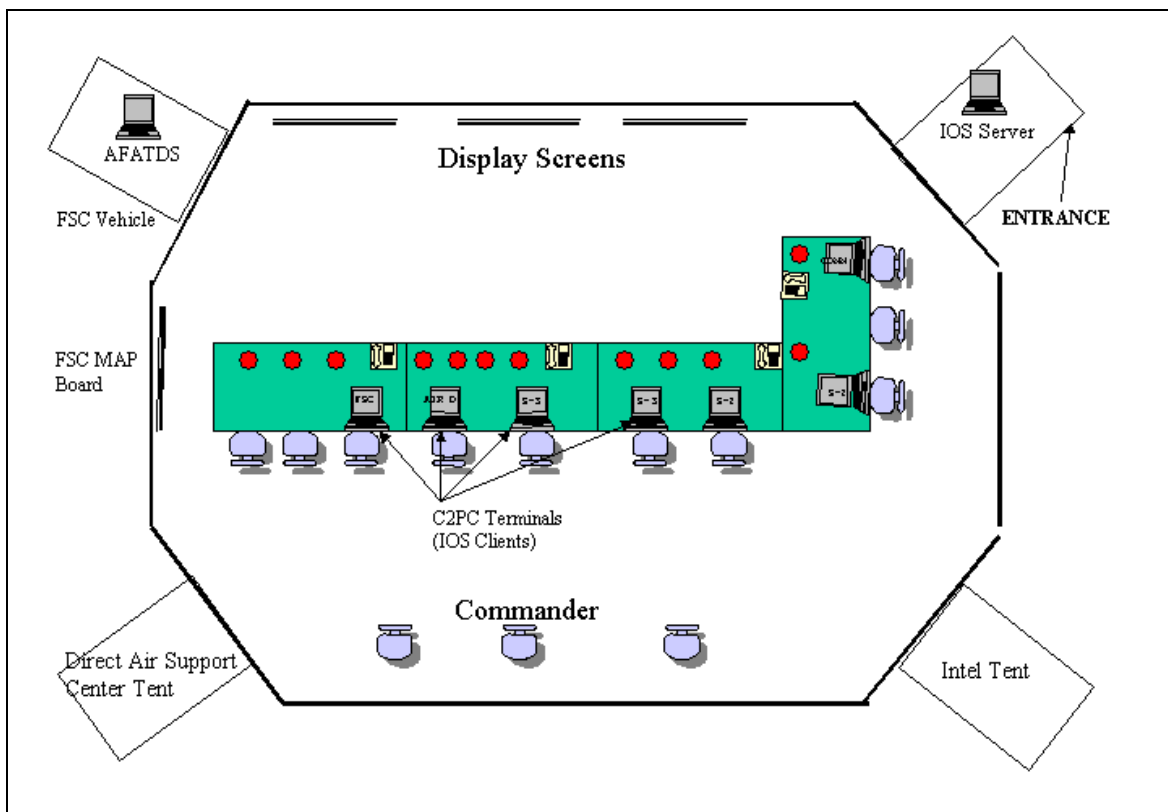


Figure 25. Example Regimental COC. (After [LITT02])

The Infantry Battalion COC is a smaller version of the Regimental COC. Typically, the Battalion COC does not have the large display screens or more than 3 C2PC terminals, and does not have an IOS server. The infantry Battalions also do not have the tentage to support a large COC, and generally run a COC out of the back of HUMMWV's or in one or two AAVC7's (Assault Amphibious Vehicle – C2 variant). The infantry battalion does have 2 AFATDS systems, brought by the Artillery Liaison Officer when he is attached.

The Artillery Battalion “COC” is named the Fire Direction Center (FDC), and contains 3 AFATDS systems. The Battalion FDC serves as the clearinghouse for fire missions from the Observers attached to the Regiment. The Battalion FDC then tasks fire missions to the Artillery Batteries. The artillery Batteries have two AFATDS terminals, with one terminal providing technical fire direction.

Digital connectivity between the Regiment and the four Battalions (this includes the Artillery Battalion) is maintained by either an EPLRS network or SINCGARS radio network. SINCGARS will support up to a 10 kbps connection, while EPLRS will support up to 57 kbps, with an upgrade to 276 kbps expected by the end of 2002.

Although AFATDS and IOS both use the same IP-based wireless communications protocol and can be routed and multiplexed, the 6th Marines communications officer had to separate AFATDS traffic from IOS traffic onto different networks due to unspecified incompatibilities. IOS traffic carried the Regimental COP, and used the EPLRS network, while AFATDS traffic traveled on a SINCGARS network. [MARC02]

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California
3. Marine Corps Representative
Naval Postgraduate School
Monterey, California
4. Director, Training and Education, MCCDC, Code C46
Quantico, Virginia
5. Director, Marine Corps Research Center, MCCDC, Code C40RC
Quantico, Virginia
6. Marine Corps Tactical Systems Support Activity (Attn: Operations Officer)
Camp Pendleton, California